Бюджетное учреждение
дополнительного профессионального образования
Ханты-Мансийского автономного округа – Югры
«Институт развития образования»

**А.В. Алексеев, С.Н. Беляев**

# ПОДГОТОВКА ШКОЛЬНИКОВ К ОЛИМПИАДАМ ПО ИНФОРМАТИКЕ С ИСПОЛЬЗОВАНИЕМ ВЕБ-САЙТА

Учебно-методическое пособие
для учащихся 7-11 классов

Ханты-Мансийск
2008

ББК 74.263.2
А 47

*Рекомендовано решением Ученого совета*
*БУ ДПО Ханты-Мансийского автономного округа – Югры*
*«Институт развития образования» от 30.05.2008 г.*
*Протокол № 6*

Рецензенты: **Т.Д. Карминская**, к.т.н., доцент, заместитель директора Департамента образования и науки Ханты-Мансийского автономного округа – Югры
**В.А. Вьюн**, д.ф.-м.н., заведующий кафедрой естественно-математического образования бюджетного учреждения дополнительного профессионального образования Ханты-Мансийского автономного округа – Югры «Институт развития образования»

**Алексеев А.В.**

А 47 **Подготовка школьников к олимпиадам по информатике с использованием веб-сайта**: учебно-методическое пособие для учащихся 7-11 классов. / А.В. Алексеев, С.Н. Беляев. – Ханты-Мансийск: РИО ИРО, 2008. – 284 с.
ISBN

В пособии рассмотрена технология дистанционной подготовки школьников к олимпиадам по информатике. Предлагается это делать с использованием сайта в сети Интернет. Приведены материалы сайта и методические рекомендации по работе с ним. Пособие ориентировано на школьников 7-11 классов, студентов вузов, школьных учителей информатики и преподавателей программирования различных учебных заведений дополнительного и профессионального образования.

ББК 74.263.2

ISBN

# Содержание

# Введение

В работе рассматривается один из способов подготовки школьников и учителей к олимпиадам по информатике. Предлагается для этого использовать ресурсы проекта Красноярского Дворца пионеров и школьников «Школа программиста», представленного на сайте http://acm.dvpion.ru (с апреля 2008 года сайт доступен также по адресу http://acmp.ru).

Сайт создан для развития у школьников навыков программирования и способностей, направленных на решение олимпиадных задач. Надеемся, что этот проект найдет массу заинтересованных в этой области людей, желающих оказать поддержку в развитии олимпиадного и спортивного программирования, олимпиадной информатики.

Сайт содержит архив задач по олимпиадному программированию со встроенной проверяющей системой. Для участия в системе достаточно зарегистрироваться и перейти в раздел «Архив задач», где на текущий момент Вам будет предложено решить 350 задач различной сложности. Сложность задач определяется числом от 1 до 100, из этих значений сложности формируется рейтинг, отражаемый в разделе «Рейтинг».

Уровень представленных здесь задач значительно легче задач, расположенных на подобных площадках. Поэтому большинство задач возможно решить после одного года обучения программированию. Средняя сложность задач немного ниже уровня задач районных олимпиад, что позволяет успешно готовиться к ним, решая задачи здесь. Сайт ориентирован на преимущественно школьную аудиторию и представляет малый интерес для студентов вузов, занимающихся олимпиадным программированием и имеющих больший опыт. В последствии в процессе роста навыков программирования предполагается плавный переход с этой системы на более серьезные, например, Уральского государственного университета (http://acm.timus.ru), где представлены более серьезные задачи. Эту систему можно рассматривать как базовую в подготовке к решению задач по программированию.

Используемые в системе задачи взяты из различных источников: районные и краевые (окружные) олимпиады по информатике, многие задачи составлены разработчиками самостоятельно, часть задач взята с сайта Интернет-олимпиад по программированию г. Санкт-Петербурга (http://neerc.ifmo.ru/school/io), а также многих других сайтов.

В помощь школьнику на сайте представлен раздел «Курс олимпиадника», в котором разобран набор базовых тем олимпиадной информатики. Для каждой из задач этого раздела приведен ее разбор в разделе «Архив задач».

Сайт разработан в Красноярском краевом Дворце пионеров и школьников при поддержке Красноярского регионального координационного центра проекта «Информатизация системы образования» Национального фонда подготовки кадров. В реализации этого проекта участвует Институт развития образования как региональный координационный центр по Ханты-Мансийскому автономному округу – Югре.

С использованием этого сайта в 2007-2008 учебном году были проведены в экспериментальном режиме школьные, муниципальные и окружная олимпиады для учащихся Ханты-Мансийского автономного округа – Югры, Интернет-турнир школ по программированию. В пособии приведены задачи этих олимпиад с разборами и программами на алгоритмическом языке Паскаль. Также этот сайт используется в работе курсов повышения квалификации учителей информатики «Методика и содержание подготовки школьников к олимпиадам по информатике», проводимых Институтом развития образования, факультативной работе с учениками Югорского физико-математического лицея и студентами Института прикладной математики, информатики и управления Югорского государственного университета.

В конце пособия приведены список сайтов с тестирующими системами и список литературы по олимпиадной информатике, доступной через Интернет-магазины и в электронном виде.

Авторы пособия надеются, что сайт и само пособие будут постоянно развиваться. Свои предложения и замечания просим направлять нам электронной почтой (aav@uriit.ru и/или bsn@mail.ru). Последние изменения в пособии будут отражаться на соответствующей странице сайта проекта «Задача в неделю» http://zvn.irohmao.ru.

# Олимпиадная информатика

Этот раздел предназначен для пользователей, начинающих осваивать программирование и служит ознакомительным аспектом с областью олимпиадной информатики. Если Вы пока не знаете что из себя представляют олимпиадные задачи, как они представлены и какие критерии оценки для проверки решения существуют, то этот раздел именно для Вас.

В отличии от обычных программ, создаваемых программистами повседневно, класс олимпиадных задач достаточно узок, но практичен с точки критериев выявления способности участников программировать за короткий срок. Как правило, олимпиадная задача представляет собой некоторую проблему, для решения которой требуется использовать свой IQ почти на пределе, однако, сам текст программы может быть совсем незначительным и помещаться на одной странице.

Если человек не занимался программированием, то предположительно можно оценить его способности к этой области в случае ее изучения. Многие полагают, что способности программировать связаны с умением решать математические и комбинаторные задачи. Другими словами, если у Вас в школе твердая пятерка по алгебре, геометрии и иным математическим дисциплинам, а так же умеете хорошо играть в шашки и шахматы, то вполне вероятно, что будете неплохо программировать, если начнете этим заниматься. И наоборот, если в школе у Вас тройка по алгебре, как бы вы не старались, то вряд ли программирование - это то, чем Вам стоит заниматься. Так же следует отметить, что Ваши заслуги в области освоения гуманитарных предметов мало Вам помогут в освоении программирования, которое, как Вы уже поняли, относится к точным наукам.

Приведем условную классификацию олимпиадных задач:

- *Арифметика* – математические задачи, работа с большими числами (длинная арифметика), такие задачи, как правило, требуют знания формул, умение их применять, а код программ может быть небольшим;
- *Геометрия* – геометрические задачи, здесь может быть описана какая либо ситуация взаимодействия тел на плоскости и в пространстве;
- *Динамическое программирование* – задачи, направленные на выявление рекуррентных соотношений;
- *Сортировка и последовательности* – работа с данными, представленными в виде массива;
- *Графы* – задачи с графами (структурами данных, основанных на вершинах и ребрах);
- *Рекурсия* – задачи на поиск с рекурсивным перебором вариантов.

Конечно, задачи могут сочетать в себе сразу несколько направлений и часто бывает сложно конкретную задачу отнести к тому или иному разделу.

Любая олимпиадная задача подразумевает входные и выходные данные. Т.е. в формулировке задания обязательным образом описан формат входных и выходных данных, а Ваша программа должна считать эти данные, обработать и вывести результат в установленном формате. Чаще всего чтение происходит из некоторого файла INPUT.TXT, а вывод в некоторый файл OUTPUT.TXT. Т.е. для решения олимпиадных задач нужно уметь работать с файлами: читать, создавать и писать в них, а вот знания графических функций вряд ли Вам пригодятся. Стоит заметить, что многие системы, например http://acm.timus.ru, используют консольный режим ввода-вывода и работа с файлами в них не приветствуется. Помимо условия задачи, правил ввода и вывода информации на каждую задачу накладываются ограничения на время выполнения и используемую Вашей программой оперативную память.

Приведем пример формулировки олимпиадной задачи по программированию (Задача № 1 в тестирующей системе из раздела «Архив задач»):

## A+B

*(Время: 1 сек. Память: 16 Мб Сложность: 2%)*

Требуется сложить два целых числа A и B.

**Входные данные**

В единственной строке входного файла INPUT.TXT записано два натуральных числа через пробел, не превышающих $10^9$.

**Выходные данные**

В единственную строку выходного файла OUTPUT.TXT нужно вывести одно целое число – сумму чисел A и B.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 2 3 | 5 |

Эта классическая простая задача используется для ознакомления участников с системой автоматической проверки и соответствует всем критериям правильной постановки олимпиадной задачи. При решении этой задачи необходимо из входного файла INPUT.TXT, расположенного в текущей папке (где и Ваша программа), считать 2 целых числа и вывести их сумму в выходной файл OUTPUT.TXT. Ограничения по памяти в 16 Мб и времени 1 сек. весьма условны, так как такая простая задача потребует минимальную память и выполнится за минимальный промежуток времени (операция сложения выполнится мгновенно, современные ЭВМ способны выполнять $10^8$ таких операций в секунду). Каждая задача имеет пример входных и выходных данных (часто даже несколько примеров), это позволяет участникам более однозначно понять содержание задачи. В данном примере в разделе «Пример» отражен пример входных данных «2 3» и выходных «5», это означает, что 2+3=5.

В зависимости от правил соревнований или тестирующей системы могут использоваться те или иные языки программирования. Наиболее часто используемыми средствами создания программ для решения олимпиадных задач являются:

- Turbo Pascal 7.1
- Borland C++ 3.1
- Borland Delphi 7.0
- Borland C++ Builder 5.0
- Microsoft Visual C/C++ 7.1
- Java 2 SDK 1.5

В мире предпочтение отдается языку C++, но в России по-прежнему классическим языком программирования остается Паскаль, а именно, большинство олимпиадных задач в России решается на Delphi. Мы же рекомендуем осваивать язык C++, который со временем станет наиболее популярным и в нашей стране.

Приведем пример решения рассмотренной выше задачи о сложении двух чисел на языках программирования, допустимых для работы в системе:

```
//Реализация задачи №1 "A+B" на C
#include < stdio.h >

long a,b;

int main(){
  freopen("input.txt","r",stdin);
  freopen("output.txt","w",stdout);
  scanf("%ld%ld",&a,&b);
  printf("%ld",a+b);
  return 0;
}
```

```
{Реализация задачи №1 "A+B" на Pascal}
var a, b : longint;

begin
  assign(input, 'input.txt'); reset(input);
  assign(output, 'output.txt'); rewrite(output);
  read(a, b);
  write(a + b);
end.
```

```
//Реализация задачи №1 "A+B" на Java
import java.util.*;
import java.io.*;

public class Main{ //имя класса должно быть Main
  public static void main(String[] argv) throws IOException{
    new Main().run();
  }
  PrintWriter pw;
  Scanner sc;
  public void run() throws IOException{
    sc = new Scanner(new File("input.txt"));
    int a=sc.nextInt(), b=sc.nextInt();
    pw = new PrintWriter(new File("output.txt"));
    pw.print(a+b);
    pw.close();
  }
}
```

```
'Реализация задачи №1 "A+B" на Basic
open "input.txt" for input as #1
open "output.txt" for output as #2
input #1,a#,b#
print #2,a#+b#
close #1
close #2
```

Теперь, ознакомившись с работой системы, Вы можете сдать данную задачу на сайте.

# Работа в системе

Сайт acm.dvpion.ru содержит архив задач по олимпиадному программированию со встроенной проверяющей системой. Для работы в системе достаточно зарегистрироваться и перейти в раздел «Архив задач», где на текущий момент Вам предлагается решить 350 задач различной сложности. Сложность задач определяется числом от 1 до 100, из этих значений сложности формируется рейтинг, отражаемый в разделе «Рейтинг» для каждого участника.

Все задачи требуют работы с файлами INPUT.TXT и OUTPUT.TXT, предназначенными для чтения входных данных и вывода результата соответственно. Отправлять решения можно только зарегистрированным пользователям в виде исходного кода в файлах с расширениями *.pas, *.dpr, *.c, *.cpp. Проверяющая система обрабатывает только программы, реализованные на языках Pascal, C++, Java и Basic, используя следующие компиляторы:

- Borland Delphi 6.0
- Microsoft Visual C++ 6.0
- Java 2 SDK 1.5
- Microsoft QBasic 4.5

Каждое отправленное решение проходит на сервере проверку не менее чем на 10 тестах, специально составленных для анализа Ваших решений. Задача считается решенной только в случае прохождения всех тестов. В случае неверного решения процесс тестирования прерывается на тесте, определившем ошибку. Результаты тестирования Ваших задач

можно видеть в разделе «Состояние системы», так же там отображаются результаты сдач других пользователей. Результатом проверки является итоговое сообщение системы и номер теста, вызвавшего ошибку (если таковая имела место).

Возможные типы сообщений могут быть представлены в виде следующей таблицы:

| № | **Сообщение** | **Событие** | **Причина** |
|---|---|---|---|
| 1 | Accepted | Программа работает правильно и прошла все необходимые тесты с соблюдением всех ограничений | |
| 2 | Wrong answer | Неверный ответ. Результат работы программы не совпадает с ответом жюри | Неверный формат вывода или алгоритмическая ошибка в программе |
| 3 | Time limit exceeded | Превышен указанный в задаче лимит времени. Программа выполняется дольше установленного времени | Неэффективное решение или алгоритмическая ошибка в программе |
| 4 | Presentation Error | Отсутствие выходного файла OUTPUT.TXT | Файл не создан, неверное имя файла или сбой программы до открытия выходного файла |
| 5 | Compilation error | Ошибка компиляции. В результате компиляции не создан исполняемый файл | Синтаксическая ошибка в программе или неверно указано расширение файла |
| 6 | Memory limit exceeded | Превышен указанный в задаче лимит памяти. Программа использует больше установленного размера памяти. | Неэффективный алгоритм, либо нерациональное использовании памяти |
| 7 | Runtime error | Ошибка исполнения. Программа завершила работу с ненулевым кодом возврата. В этом случае результат работы не проверяется | Возможно, в программе произошло обращение к несуществующему элементу массива, деление на ноль и т.д. Возможно, программа на C++ не завершается оператором "return 0" или по иной причине вернула ненулевой код возврата |
| 8 | Compiling | Компиляция программы | Необходимо время для создания исполняемого файла |
| 9 | Running | Исполнение программы | Идет тестирование программы путем ее запуска для каждого имеющегося теста |
| 10 | Waiting | Ожидание | Программа находится в очереди тестируемых программ, либо не работает проверяющая система |

Если Вы еще не сдавали задачи на этом сайте, но уже горите желанием, то рекомендуем Вам начать со сдачи задачи № 1 «A+B» для понимания процесса работы проверяющей системы. Пример решения данной задачи уже был приведен в разделе «Олимпиадная информатика» на языках Pascal и C. Удачи!!!

# Курс начинающего олимпиадника

Данный раздел содержит набор тем для самостоятельного изучения основ курса олимпиадного программирования. Каждая тема включает теоретическую часть и набор задач, сложность которых растет от темы к теме. Для рассмотренных в этом разделе задач в разделе «Архив задач» приведено словесное описание их решений. Но все же мы рекомендуем вам сначала попытаться их решить самостоятельно, и лишь в том случае, когда это сделать не получается, прибегать к просмотру их решения.

Этот курс предназначен для школьников, имеющих некоторые знания в области программирования на уровне понимания синтаксиса языка и простейших алгоритмов.

## Список тем

**Тема 01:** Введение. Понятие олимпиадной задачи. Работа с файлами.
**Тема 02:** Разветвляющиеся алгоритмы.
**Тема 03:** Циклические алгоритмы.
**Тема 04:** Массивы.
**Тема 05:** Сортировка.
**Тема 06:** Двумерные массивы.
**Тема 07:** Системы счисления.
**Тема 08:** Теория чисел.
**Тема 09:** Длинная арифметика.

## **Тема 01:** Введение. Понятие олимпиадной задачи. Работа с файлами.

При решении олимпиадных задач необходимо уметь работать с текстовыми файлами, т.к. от участника олимпиады, как правило, требуется написать программу, которая считывает некоторые данные из одного файла, производит определенные вычисления, а результат выводит в другой файл.

Для работы с файлами как в языке Паскаль, так и в языке Си можно обойтись без использования файловых переменных. Добавив две строчки кода в программу, можно перенаправить ввод данных с консоли на ввод из файла, а вывод на экран заменить на вывод в файл. Следующие фрагменты кода реализуют данную возможность:

```
// Язык Си
freopen("input.txt","r",stdin);
freopen("output.txt","w",stdout);
{Язык Паскаль}
assign(input, 'input.txt'); reset(input);
assign(output, 'output.txt'); rewrite(output);
```

Перед выполнением заданий данного раздела рекомендуем ознакомиться с разделом «Олимпиадная информатика», в котором дана классификация олимпиадных задач, разобрана структура олимпиадной задачи и приведены примеры работы с файлами на языках Паскаль и Си. С особенностями работы автоматизированной системы проверки и видами возможных ошибок можно ознакомиться в разделе «Работа в системе».

В этой теме мы предлагаем решить простейшие задачи, которые помогут ознакомиться с вводом-выводом данных. Подобные задачи обычно используются на пробных турах олимпиад по программированию.

## Список задач

**Задача 001:** A+B.
**Задача 108:** Неглухой телефон.
**Задача 033:** Два бандита.
**Задача 092:** Журавлики.
**Задача 004:** Игра.

## **Тема 02:** Разветвляющиеся алгоритмы.

Мы полагаем, что Вы уже сталкивались с условным оператором и понимаете, как он работает. Хотелось бы, чтобы в этом разделе Вы закрепили свои знания на примере решения простых олимпиадных задач с использованием условий.

Напомним так же, что в языке Си наряду с условным оператором существуют так же операторы switch и «?», а в языке Pascal – оператор case. Например, в языке Си при нахождении максимального элемента из двух чисел вместо следующего условного оператора:

```
if(a > b) max=a; else max=b;
```

можно использовать следующий вариант записи:

```
max=(a>b)?a:b;
```

### Список задач

**Задача 025:** Больше – меньше.
**Задача 021:** Зарплата.
**Задача 008:** Арифметика.
**Задача 052:** Счастливый билет.
**Задача 026:** Две окружности.

## **Тема 03:** Циклические алгоритмы.

Сложно представить серьезную программу без циклов. Мы полагаем, что вы уже знакомы с циклами. В этом разделе будет рассмотрен ряд задач, для решения которых необходимо использовать циклы.

Напомним, что существуют три вида циклов:

1. Оператор цикла с параметром

```
for i=1..n{
  Операторы;
}
```

2. Оператор цикла с предусловием

```
while(Условие){
  Операторы;
}
```

3. Оператор цикла с постусловием

```
do{
  Операторы;
}while(Условие);
```

### Список задач

**Задача 106:** Монетки.
**Задача 081:** Арбузы.
**Задача 043:** Нули.
**Задача 063:** Загадка.
**Задача 002:** Сумма.

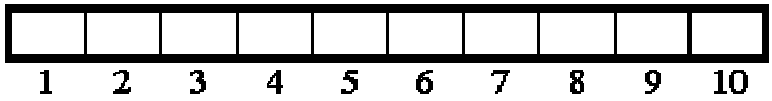
## **Тема 04:** Массивы.

Массив – это упорядоченная совокупность переменных, объединенных общим типом и именем. Число элементов массива фиксируется при описании и в процессе выполнения программы не меняется. Каждый элемент массива определяется именем, совпадающим с именем массива, а также индексом. Индекс – это величина, характеризующая положение элемента в массиве.

В языке Си нумерация элементов массива начинается с нуля, таким образом номер последнего элемента массива соответствует значению n-1, где n – количество элементов массива. В Паскале при описания массива можно определять диапазон индексов, что более удобно. Приведем пример описания целочисленного массива на разных языках:

```
// Язык Си, элементы массива имеют номера от 0 до 9
int a[10];
{Язык Паскаль, нумерация элементов от 1 до 10}
var a : array [1..10] of integer;
//Алгоритмический язык этого курса, нумерация от 1 до 10
int a[1..10];
```

Схематическое изображение массива из 10 элементов:




К каждому элементу массива можно обращаться отдельно так же как к обычной переменной установленного типа. Элемент массива с номером i имеет обозначение a[i]. Благодаря такой записи данные можно обрабатывать массово в цикле. Входные данные задачи, которые необходимо хранить в массиве, обычно задаются следующим образом: сначала идет число N – количество элементов массива, а за ним следуют отделенные пробелом значения N элементов массива. Алгоритм чтения N элементов массива и вывод их в файл может выглядеть следующим образом:

```
const MaxN=100;
int i,n,a[1..MaxN];

read(n);
//чтение элементов из файла в массив
for i=1..n{
  read(a[i]);
}
//вывод элементов из массива в файл
for i=1..n{
  write(a[i],' ');
}
```

### Список задач

**Задача 149:** Разворот.
**Задача 082:** Пересечение множеств.
**Задача 039:** Волосатый бизнес.

## **Тема 05:** Сортировка.

Сортировка – преобразование последовательности элементов в неубывающую (или невозрастающую) последовательность. Последовательность элементов {Ai} называют неубывающей, если для любых i и j, таких что i < j выполняется неравенство Ai $\leq$ Aj. Для строго возрастающей последовательности неравенство принимает вид Ai < Aj. Аналогичным образом определяется невозрастающая и убывающая последовательности.


При решении олимпиадных задач часто необходима сортировка данных. Обычно данные представлены в виде массива из N элементов, где элементы – это чаще всего числа или строки. Для этого существует множество алгоритмов, которые с помощью замены значений элементов исходного массива приводят его к отсортированному виду.

Для лучшего понимания того, в каких случаях нужно применить тот или иной алгоритм необходимо знать, что понимают под показателем сложности алгоритма. Речь идет о том, как зависит число операций, которые нужно произвести согласно алгоритму от объема данных, в нашем случае от количества элементов массива N. Сложность задачи может быть логарифмической, линейной, квадратичной, экспотенциальной и т.д., где для решения задачи необходимо выполнение O(ln(N)), O(N), O($N^2$) и O($e^N$) операций соответственно. Например, квадратичный порядок сложности O($N^2$) означает, что задача может использовать $N^2$ операций, а может и 100*$N^2$, здесь коэффициент перед $N^2$ не имеет значения: важен порядок, важно знать во сколько раз программа будет работать дольше, если число N увеличится вдвое, втрое или в 10 раз. В нашем случае независимо от этого коэффициента получим, что программа будет выполняться соответственно в 4, 9 и 100 раз дольше.

Наилучшие универсальные алгоритмы сортировки имеют порядок сложности O(n*ln(n)), что позволяет в олимпиадных задачах сортировать массивы для N = 300 000 (приблизительно). В качестве примера подобных алгоритмов можно привести следующие сортировки: быстрая, пирамидальная, слиянием, бинарным деревом.

Простейшие алгоритмы сортировки имеют порядок $O(N^2)$, что позволяет решать задачу с сортировкой только для $N \leq 5000$ (так же примерно). Несмотря на то, что квадратичные алгоритмы дают меньший эффект, их разумно использовать, когда скоростью сортировки данных можно пренебречь, повысив скорость реализации программы. Примеры подобных сортировок: выбором, пузырьком, вставками.

В частных случаях отсортировать данные возможно с линейным порядком сложности, когда есть некоторые ограничения на данные (например, массив уже частично отсортирован или диапазон элементов массива ограничен). С таким порядком сложности возможна сортировка массива, состоящего из $10^7$ элементов! Например, следующие алгоритмы дают такой результат: цифровая и поразрядная сортировки.

Многие считают, что достаточно знания двух сортировок (например, «пузырьком» и «быстрой»), чтобы решать любые олимпиадные задачи. Но на самом деле это далеко не так. Далее, рассмотрим алгоритмы сортировки, наиболее часто используемые программистами. Во всех примерах будем сортировать по неубыванию полагая, что a[i] – целочисленный массив, состоящий из n элементов, с индексами от 1 до n.

## Сортировка выбором

Пожалуй, это самый легкий для реализации алгоритм среди существующих, идея которого сводится к последовательному позицированию нужных элементов с 1-го по n-ый. Вначале среди всех элементов определяется наименьший и меняется с 1-ым, далее среди всех оставшихся снова находится наименьший и меняется со 2-ым. Далее, среди элементов, начиная с 3-го так же находится наименьший и меняется с 3-им и т.д. до (n-1)-го элемента. Квадратичная сложность алгоритма очевидна, а алгоритм решения может выглядеть следующим образом:

```
for i=1..n-1{
  for j=i+1..n{
    if(a[i] > a[j]) a[i] <---> a[j];
  }
}
```

## Сортировка пузырьком

Это, наверное, самый популярный алгоритм сортировки, идея которого чем то похожа на предыдущий алгоритм: так же реализация сводится к позицированию нужных элементов с 1-го по последний. Сама процедура установки текущего элемента в нужную позицию чем то напоминает всплытие пузырька. Для того, чтобы первый элемент (наименьший) встал на свое место необходимо справа налево пройтись по массиву, попарно сравнивая соседние элементы и в том случае, когда левый больше правого менять их местами. Для позицирования второго элемента, нужно пройтись еще раз по массиву, но не до 1-го а уже до 2-го элемента и т.д. Данный алгоритм, который как и предыдущий имеет квадратичную сложность, можно представить следующим образом:

```
for i=1..n-1{
  for j=n-1..i{
    if(a[j] > a[j+1]) a[j] <---> a[j+1];
  }
}
```

## Быстрая сортировка

Этот алгоритм является одним из самых популярных и наиболее часто используемым среди алгоритмов, имеющих порядок сложности O(n*ln(n)). Сам алгоритм является рекурсивным и его идея заключается в следующем: для сортировки массива достаточно разбить его на две части так, чтобы все элементы левой части были меньше либо равны всех элементов правой части, далее следует выполнить подобную операцию с левой и правой частью,

рассматривая их как отдельные массивы. Алгоритмическое представление процедуры, которая сортирует подмассив с l-го по r-й элемент можно представить следующим образом:

```
sub QuickSort(l,r){
  x=(l+r) div 2;
  i=l; j=r;
  do{
    while(a[i] < p) i=i+1;
    while(a[j] > p) j=j-1;
    if (i <= j) {
      a[i] <---> a[j];
      i=i+1; j=j+1;
    }
  }while(i <= j);
  if(j > l) QuickSort(l, j);
  if(r > i) QuickSort(i, r);
}
```

Для сортировки массива достаточно вызвать процедуру QuickSort(1, n).

### Список задач

**Задача 119:** Сортировка времени.
**Задача 032:** Годовой баланс.
**Задача 041:** Цифровая сортировка.

## **Тема 06:** Двумерные массивы

**Двумерный массив** – это одномерный массив, элементами которого являются одномерные массивы. Другими словами, это набор однотипных данных, имеющий общее имя, доступ к элементам которого осуществляется по двум индексам. Наглядно двумерный массив удобно представлять в виде таблицы, в которой n строк и m столбцов, а под ячейкой таблицы, стоящей в i-й строке и j-м столбце понимают некоторый элемент массива a[i][j].

Действительно, если разобраться с тем, что такое a[i] при фиксированном значении i, то увидим, что это одномерный массив, состоящий из m элементов, к которым можно обращаться по индексу: a[i][1], a[i][2], ... , a[i][m]. Схематически это вся i-я строка строка таблицы. Аналогично, если мы рассмотрим одномерный массив строк, то сможем заметить, что это так же двумерный массив, где каждый отдельный элемент – это символ типа char, а a[i] – это одномерный массив, представляющий отдельную строку исходного одномерного массива строк. Исходя из идеи определения двумерного массива, можно определить рекуррентное понятие многомерного массива:

**n-мерный массив** – это одномерный массив, элементами которого являются (n-1)-мерные массивы.

Несложно догадаться, что 3-мерный массив визуально можно представить в виде куба с ячейками (похоже на кубик Рубика), где каждый элемент имеет вид a[i][j][k]. А вот с большими размерностями возникают сложности с визуальным представлением, но математическая модель ясна.

По-другому двумерный массив также называют **матрицей**, а в том случае, когда n=m (число строк равно числу столбцов) матрицу называют квадратной. В матрицах можно хранить любые табличные данные: содержание игрового поля (шашки, шахматы, Lines и т.д.), лабиринты, таблицу смежности графа, коэффициенты системы линейных уравнений и т.д. Матрицы часто используют для решения олимпиадных и математических задач.

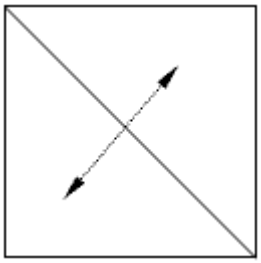
В задачах табличные данные часто определяются во входном файле следующим образом: сначала в первой строке указываются значения n и m через пробел, а далее идут n строк по m элементов в каждой, также друг от друга отделенные пробелом и входной файл может иметь, например, следующее содержание, понятно отражающее содержимое матрицы при обычном просмотре:

```
3 5
7 8 2 3 1
5 3 2 6 3
9 3 5 2 0
```

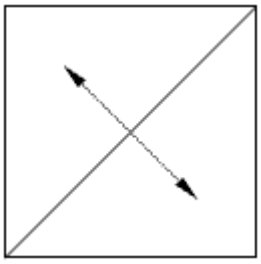
В приведенном примере определена матрица, состоящая из трех строк и пяти столбцов. Рассмотрим пример чтения этих данных в матрицу и вывода матрицы в файл. Для этого удобно использовать двойной цикл, где внешний цикл по i будет пробегать по всем строкам, а внутренний цикл по j будет для текущей строки i перебирать все ее элементы. Алгоритмическая реализация этого процесса может выглядеть следующим образом:

```
int a[1..100][1..100];
//Чтение данных двумерного массива из файла
read(n,m);
for i=1..n{
  for j=1..m{
    read(a[i][j]);
  }
}
//Вывод матрицы в файл
for i=1..n{
  for j=1..m{
    write(a[i][j], ' ');
  }
  writeln; //новая строка
}
```

В качестве примера рассмотрим задачу о транспонировании квадратной матрицы относительно главной и побочной диагонали, где необходимо симметричным образом поменять элементы двумерного массива относительно одной из диагонали. Алгоритмическое решение данной задачи может быть представлено следующим образом:



```
for i=2..n{
  for j=1..i-1{
    a[i][j] <---> a[j][i];
  }
}
```



```
for i=1..n-1{
  for j=1..n-i{
    a[i][j] <---> a[n-j+1][n-i+1];
  }
}
```

## Список задач

**Задача 027:** Художник.
**Задача 058:** Проверка на симпатичность.
**Задача 088:** Судоку.

## **Тема 07:** Системы счисления

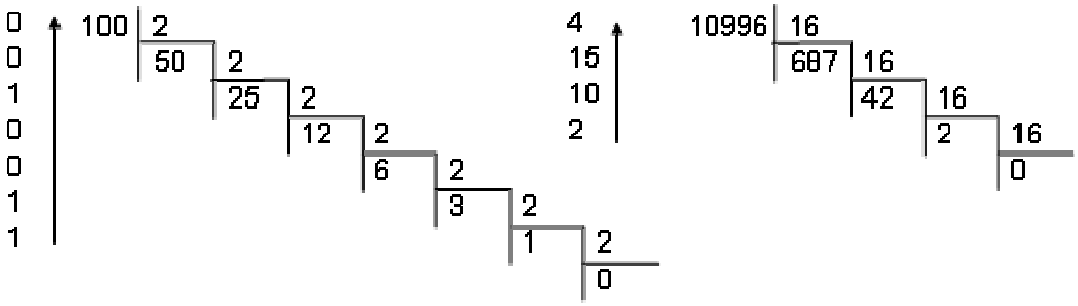
Обычно мы имеем дело с десятичной системой счисления, в которой используется ровно 10 цифр: 0..9, и мы все к этому сильно привыкли. Но для обозначения чисел возможно использование других систем, отличных от десятичной. Особенностью таких систем является, отличное от 10 количество используемых цифр. Например, в двоичной системе используется только две цифры - 0 и 1; и любое число может быть представлено в такой системе. Если система имеет порядок, больший чем 10, то в качестве следующих цифр используются латинские буквы; например, в шестнадцатеричной системе присутствуют следующие цифры: 0..9, A, B, C, D, E и F.

Для того чтобы представить какое либо число в десятичной системе счисления, следует вычислить сумму всех цифр данного числа, умноженных на порядок системы счисления, возведенной в степень, равную номеру разряда данной цифры. Следующие примеры демонстрируют суть данного метода:

$$(1100100)_2 = 1\cdot 2^6+1\cdot 2^5+0\cdot 2^4+0\cdot 2^3+1\cdot 2^2+0\cdot 2^1+0\cdot 2^0 = (100)_{10}$$
$$6\,5\,4\,3\,2\,1\,0$$

$$(2AF4)_{16} = 2\cdot 16^3+10\cdot 16^2+15\cdot 16^1+4\cdot 16^0 = (10996)_{10}$$
$$3\,2\,1\,0$$

Для перевода из десятичной системы счисления в систему с основанием k необходимо производить деление исходного числа на k, запоминая остатки от деления и продолжая данную операцию с целочисленным значением деления до тех пор, пока результатом деления не будет ноль. Искомым представлением числа будет запись, составленная из полученных остатков от деления, записанных в обратном порядке. Ниже представлены примеры перевода:




При решении олимпиадных задач обычно рассматривают системы счисления с основаниями от 2 до 36. Это связано с тем, что в латинском алфавите 26 букв (10 цифр + 26 букв = 36 символов). Для хранения цифр числа можно использовать как целочисленный массив, так и строку. Пусть S – число, записанное в K-ой системе счисления, а X – число в десятичной системе. Тогда алгоритм перевода из любой системы в десятичную можно записать следующим образом:

```
read(S);
X=0;
for i = 1..len(S){
  X = X*K+S[i];
}
write(X);
```

Алгоритм перевода из десятичной системы числа X в систему с основанием K и записи его в S может быть представлен так:

```
read(X);
l=0;
do{
  l=l+1;
  S[l] = X mod K;
  X = X div K;
}while(X>0);
write(reverse(S));
```

Для того чтобы перевести из k-ой системы счисления в систему с основанием m, можно воспользоваться «принципом чайника» и перевести сначала из k-ой системы в 10-ую, а затем из 10-ой в m-ую. Однако иногда этой двойной операции можно избежать. Речь идет о случае, когда мы имеем дело с кратными системами счисления. Например, несложно переводить из 2-ой в 4-ую, 8-ую, 16-ную и обратно; аналогично можно сказать о 3-ой и 27-ой системах или о 5-ой и 25-ой. В подобных случаях каждая цифра числа в системе с большим основанием представляется в виде нескольких цифр системы с меньшим основанием. Так, в 4-ой системе каждая цифра представляется 2-мя цифрами 2-ой системы, в 8-ой – 3мя цифрами 2-ой, а в 16-ой каждая цифра есть ничто иное как 4 цифры 2-ой системы. Например, для перевода двоичного числа 1011001010100110 11 в 16-ую систему достаточно разбить число на 4-ки:

0010 1100 1010 1001 1011 и записать каждую из них в 16-ом формате: 2CA9B, т.к. 0010 = 2, 1100 = B, 1010 = A, 1001 = 9 и 1011 = B. Несложно догадаться, что обратное преобразование из 16-ой в 2-ую систему происходит аналогичным образом.

Список задач

**Задача 022:** Единицы.
**Задача 059:** Несложное вычисление.
**Задача 275:** Делимость на 7.

## **Тема 08:** Теория чисел

*Теория чисел* – раздел математики, занимающийся изучением чисел непосредственно как таковых, их свойств и поведения в различных ситуациях. Достаточно сложно дать полное определение теории чисел, т.к. точного определения и не существует вовсе. Мы же будем рассматирать только ту ее часть, которая используется при решении олимпиадных задач. В большей степени будем уделять внимание целым числам. Для этого определим некоторые разновидности чисел ...

Множество всех *целых чисел* обычно обозначают буквой **Z** и понимают под ним набор всех действительных чисел без дробной части: {... , -3, -2, -1, 0, 1, 2, 3, ...}. *Натуральные числа* являются подмножетством целых чисел и образуют множество **N**: {1, 2, 3, ...}.

*Простым числом* называют натуральное число, большее единицы, которое делится только на 1 и на само себя. Все остальные числа называют *составными*. Первые 10 простых чисел: 2, 3, 5, 7, 11, 13, 17, 19, 23 и 29. Перечислим несколько свойств простых чисел:

- любое составное число представляется уникальным образом в виде произведения простых чисел; иначе еще говорят, что разложение числа на простые множители однозначно.
- простых чисел бесконечно много, причем существует примерно n/ln(n) простых чисел, меньших числа n.
- наименьший простой делитель составного числа n не превышает sqrt(n), поэтому для проверки простоты числа достаточно проверить его делимость на 2 и все нечетные (а еще лучше простые) числа, не превосходящие sqrt(n).
- любое четное число, большее двух представимо в виде суммы двух простых чисел; а любое нечетное, большее чем 5 представимо в виде суммы трех простых чисел
- для любого натурального n, большего единицы существует хотя бы одно простое число на интервале (n, 2*n)

*Числа Мерсена* – это числа, представимые в виде $2^n$-1. Особый интерес представляют простые числа Мерсена, которые получаются при n=2, 3, 5, 7, 13, 17, 19, 41, 47, 61, 89, 10, 127, ... На сегодняшний день известно 44 простых числа Мерсена и самое большое из них получается при n=32582657 и содержит в себе почти 10 миллионов цифр, оно же является самым большим из найденных на сегодняшний день. Это же число является наибольшим среди всех известных простых чисел. На сегодняшний день неизвестно: конечно ли число простых чисел Мерсена.

*Числа Ферма* – это числа, представимые в виде $2^{2^n}+1$. Простыми среди чисел вида $2^n$+1 могут быть только числа Ферма. На данный момент известно всего 5 простых чисел Ферма: 3, 5, 17, 257, 65537; так же известно, что для $5 \le n \le 32$ все числа Ферма – составные.

*Совершенное число* – это натуральное число, равное сумме всех своих делителей, не включая самого себя. Например, число 28 – совершенное число, т.к. 28=1+2+4+7+14. Вот первые 10 совершенных чисел: 6, 28, 496, 8128, 33550336, 8589869056, 137438691328, 2305843008139952128, 2658455991569831744654692615953842176, 191561942608236107294793378084303638130997321548169216. Известно, что любое четное совершенное число может быть представлено в виде $2^{p-1}(2^p-1)$, где число $2^p$-1 является простым числом Мерсена. На сегодняшний день не известно конечно ли количество совершенных чисел и существуют ли нечетные совершенные числа.

*Дружественные числа* – два натуральных числа, для которых сумма всех делителей первого числа (кроме него самого) равна второму числу и сумма всех делителей второго числа (кроме него самого) равна первому числу. Иногда частным случаем дружественных чисел считаются совершенные числа: каждое совершенное число дружественно себе. Обычно же, говоря о дружественных числах, имеют в виду пары из двух разных чисел. Первые 8 пар таких чисел: 220 и 284, 1184 и 1210, 2620 и 2924, 5020 и 5564, 6232 и 6368, 10744 и 10856, 12285 и 14595, 17296 и 18416.

*Число Армстронга* – натуральное число, которое равно сумме своих цифр, возведённых в степень, равную количеству его цифр. Например, $1634 = 1^4 + 6^4 + 3^4 + 4^4$. Последовательность чисел Армстронга начинается так: 1, 2, 3, 4, 5, 6, 7, 8, 9, 153, 370, 371, 407, 1634, 8208, 9474, 54748, 92727, 93084, 548834, 1741725, 4210818, 9800817, 9926315, 24678050, 24678051, 88593477, 146511208, 472335975, 534494836, 912985153, 4679307774, 32164049650, 32164049651 ... С одним из алгоритмов поиска таких чисел Вы можете ознакомиться здесь.

*m-самовлюбленное число* – натуральное число, которое равно сумме своих цифр, возведённых в степень m, где m – некоторое натуральное число. Числа Армстронга – частный случай таких чисел.

*Число Смита* – такое составное число, сумма цифр которого (в данной системе счисления) равняется сумме цифр всех его простых сомножителей. Так, примером числа Смита может служить 202, поскольку 2 + 0 + 2 = 4, и 2 + 1 + 0 + 1 = 4 (202 = 2 * 101). У. Л. МакДэниел доказал, что существует бесконечно много чисел Смита. Насчитывается 29 928 чисел Смита в пределах до 1 000 000. Первые 50 чисел Смита: 4, 22, 27, 58, 85, 94, 121, 166, 202, 265, 274, 319, 346, 355, 378, 382, 391, 438, 454, 483, 517, 526, 535, 562, 576, 588, 627, 634, 636, 645, 648, 654, 663, 666, 690, 706, 728, 729, 762, 778, 825, 852, 861, 895, 913, 915, 922, 958, 985, 1086.

### Список задач

**Задача 148:** Наибольший общий делитель.
**Задача 014:** Наименьшее общее кратное.
**Задача 147:** Числа Фибоначчи.
**Задача 349:** Простые числа.
**Задача 323:** Гипотеза Гольбаха.
**Задача 036:** Постулат Бертрана.

## **Тема 09:** Длинная арифметика

*Длинная арифметика* – раздел олимпиадного программирования, в котором рассматривается реализация действий с большими числами, не умещающихся в стандартных типах данных. На сегодняшний день единственным языком, используемым для решения олимпиадных задач и поддерживающим длинную арифметику, является Java, где все необходимые функции для работы с длинными числами встроены и можно обойтись без трудоемких реализаций.

В основном мы будем рассматривать работу с целыми числами. Для хранения длинного числа можно использовать целочисленный массив, где в качестве элемента массива будет одна цифра числа. В 1-ом элементе массива будем хранить последнюю цифру числа, во 2-ом – предпоследнюю и т.д. до последней цифры. В 0-ом элементе можно хранить общее количество цифр в числе. В простейшем случае, для хранения числа 154 достаточно будет использовать следующую запись:

```
const maxsize=100;
int a[maxsize];
a[0]=3; a[1]=4; a[2]=5; a[3]=1;
```

Элементом массива может быть не одна цифра, возможно использовать один элемент для хранения 4х цифр, т.к. мы понимаем, что на современных ЭВМ все операции как минимум 32-разрядные, поэтому время на сложение двух цифр такое же как и на сложение чисел, состоящих из 4х цифр. Поэтому часто используют порядок системы счисления base=10000 и фактически длинные числа хранятся как бы в 10000-й системе счисления, это позволяет увеличить скорость выполнения операций над ними в 3–4 раза.

Идея реализации всех необходимых операций (сложение, вычитание, умножение, деление и т.д.) основана на тех принципах, которыми мы пользуемся при расчетах на бумаге. Даже когда мы реализуем деление «в столбик», фактически мы работаем с небольшими числами, сводя более сложную задачу к набору из более простых подзадач.

Задачи, которые рассматриваются в данном разделе, представляют собой базовый набор, достаточный для формирования первоначальных навыков овладения принципами длинной арифметики. Практически все задачи на длинную арифметику требуют чтения из файла и запись в файл длинных чисел, поэтому приведем реализацию этих функций в данном разделе, а в разборе задач будем их использовать:

```
void readlong(int *a){
  int i;
  string s;
  read(s);
  a[0]=len(s);
  for i=1..len(s){
    a[i]=ord(s[i])-48;
  }
}

void writelong(int *a){
  for i=a[0]..1{
    write(a[i]);
  }
}
```

Так же часто возникает необходимость сравнивать длинные числа. Опишем следующую функцию, которая будет возвращать 0 в случае равенства чисел, -1 когда первое число меньше второго и 1 когда первое больше:

```
void complong(int *a, int *b){
  if (a[0] < b[0]) return -1;
  if (a[0] > b[0]) return 1;
  for i=a[0]..1{
    if(a[i] < b[i]) return -1;
    if(a[i] > b[i]) return 1;
  }
  return 0;
}
```

## Список задач

**Задача 103:** A+B
**Задача 143:** A-B
**Задача 144:** A*B
**Задача 145:** A div B
**Задача 172:** A mod B
**Задача 146:** Длинный корень

# Архив задач

## 001. A+B

*(Время: 1 сек. Память: 16 Мб Сложность: 2%)*

Требуется сложить два целых числа A и B.

**Входные данные**

В единственной строке входного файла INPUT.TXT записано два натуральных числа через пробел, не превышающих $10^9$.

**Выходные данные**

В единственную строку выходного файла OUTPUT.TXT нужно вывести одно целое число – сумму чисел A и B.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 2 3 | 5 |

## 002. Сумма

*(Время: 1 сек. Память: 16 Мб Сложность: 19%)*

Требуется посчитать сумму целых чисел от 1 до N.

**Входные данные**

В единственной строке входного файла INPUT.TXT записано единственное целое число N, не превышающее по абсолютной величине $10^4$.

**Выходные данные**

В единственную строку выходного файла OUTPUT.TXT нужно вывести одно целое число – сумму чисел от 1 до N.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 5 | 15 |

**Разбор**

В этой задаче можно воспользоваться формулой арифметической прогрессии: $S_n = (a_1+a_n)*n/2$. Но школьники часто забывают эту формулу и в этом случае прибегают к вычислению этой суммы с помощью цикла и решают эту задачу, используя следующий алгоритм:

```
read(n);
s=0;
for i=1..n{
  s=s+i;
}
write(s);
```

Но сложность этой задачи не в вычислении этой суммы и приведенный выше алгоритм не проходит на третьем тесте! Оказывается, здесь очень внимательно нужно прочитать условия задачи, особенно ограничения на число N. Дело в том, что это число может быть отрицательным! Учитывая это, попробуйте самостоятельно доработать и реализовать верный алгоритм решения этой задачи.

## 003. Пятью пять – двадцать пять!

*(Время: 1 сек. Память: 16 Мб Сложность: 8%)*

Вася и Петя учатся в школе в одном классе. Недавно Петя поведал Васе о хитром способе возведения в квадрат натуральных чисел, оканчивающихся на цифру 5. Теперь Вася может с легкостью возводить в квадрат двузначные (и даже некоторые трехзначные) числа, оканчивающиеся на 5. Способ заключается в следующем: для возведения в квадрат числа, оканчивающегося на 5 достаточно умножить число, полученное из исходного вычеркивани-

ем последней пятерки на следующее по порядку число, затем остается лишь приписать «25» к получившемуся результату справа. Например, для того, чтобы возвести число 125 в квадрат достаточно 12 умножить на 13 и приписать 25, т.е. приписывая к числу 12*13=156 число 25, получаем результат 15625, т.е. $125^2$=15625. Напишите программу, возводящую число, оканчивающееся на 5, в квадрат для того, чтобы Вася смог проверить свои навыки.

**Входные данные**

В единственной строке входного файла INPUT.TXT записано одно натуральное число A, оканчивающееся на цифру 5, не превышающее $4*10^5$.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите одно натуральное число - $A^2$ без лидирующих нулей.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 5 | 25 |
| 2 | 75 | 5625 |
| 3 | 4255 | 18105025 |

## 004. Игра

*(Время: 1 сек. Память: 16 Мб Сложность: 4%)*

В свободное время одноклассники Вася и Петя любят играть в различные логические игры: морской бой, крестики-нолики, шахматы, шашки и многое другое. Ребята уже испробовали и поиграли во всевозможные классические игры подобного рода, включая компьютерные. Однажды им захотелось сыграть во что-нибудь новое, но ничего подходящего найти не удалось. Тогда Петя придумал следующую игру «Угадайка»: Играют двое участников. Первый загадывает любое трехзначное число, такое что первая и последняя цифры отличаются друг от друга более чем на единицу. Далее загадавший число игрок переворачивает загаданное число, меняя первую и последнюю цифры местами, таким образом получая еще одно число. Затем из максимального из полученных двух чисел вычитается минимальное. Задача второго игрока – угадать по первой цифре полученного в результате вычитания числа само это число. Например, если Вася загадал число 487, то перестановкой первой и последней цифры он получит число 784. После чего ему придется вычесть из 784 число 487, в результате чего получится число 297, которое и должен отгадать Петя по указанной первой цифре «2», взятой из этого числа. Петя успевает лучше Васи по математике, поэтому практически всегда выигрывает в играх такого типа. Но в данном случае Петя схитрил и специально придумал такую игру, в которой он не проиграет Васе в любом случае. Дело в том, что придуманная Петей игра имеет выигрышную стратегию, которая заключается в следующем: искомое число всегда является трехзначным и вторая его цифра всегда равна девяти, а для получения значения последней достаточно отнять от девяти первую, т.е. в рассмотренном выше случае последняя цифра равна 9-2=7. Помогите Пете еще упростить процесс отгадывания числа по заданной его первой цифре, написав соответствующую программу.

**Входные данные**

В единственной строке входного файла INPUT.TXT задана единственная цифра K, соответствующая первой цифре полученного Васей в результате вычитания наименьшего загаданного Васей значения из наибольшего.

**Выходные данные**

В выходной файл OUTPUT.TXT нужно вывести значение полученной Васей разности.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 5 | 594 |
| 2 | 9 | 990 |
| 3 | 7 | 792 |

**Разбор**

В этой задаче самое сложное - это понять, что нужно сделать. Сама задача очень проста и сводится к тому, чтобы считать одну цифру x от 1 до 9, а затем вывести три цифры: x, 9 и 9-x.

Здесь самое глупое, но понятное решение – это перебор всевозможных вариантов (их всего 9) и вывод одного из возможных решений в зависимости от входных данных. Речь идет о таком например решении:

```
...
scanf("%d",&x);
if(x==1) printf("198");
if(x==2) printf("297");
if(x==3) printf("396");
if(x==4) printf("495");
if(x==5) printf("594");
if(x==6) printf("693");
if(x==7) printf("792");
if(x==8) printf("891");
if(x==9) printf("990");
...
```

Подобное решение является верным и пройдет все тесты, но оно не приветствуется в силу своей неуниверсальности. А вдруг подобная задача будет содержать не 9, а 99 или 999 вариантов, тогда перебирая их можно потерять массу времени и допустить ошибку. Поэтому полезно вывести формулу, даже если задача очень легкая и подразумевает подобное банальное решение.

Надеемся, что Вы не воспользуетесь вышеприведенным кодом для решения этой задачи.

## 005. Статистика

*(Время: 1 сек. Память: 16 Мб Сложность: 25%)*

Вася не любит английский язык, но каждый раз старается получить хотя бы четверку за четверть, чтобы оставаться ударником. В текущей четверти Вася заметил следующую закономерность: по нечетным дням месяца он получал тройки, а по четным – четверки. Так же он помнит, в какие дни он получал эти оценки. Поэтому он выписал на бумажке все эти дни для того, чтобы оценить, сколько у него троек и сколько четверок. Помогите Васе это сделать, расположив четные и нечетные числа в разных строчках. Вася может рассчитывать на оценку 4, если четверок не меньше, чем троек.

**Входные данные**

В первой строке входного файла INPUT.TXT записано единственное число N – количество элементов целочисленного массива ($1 \le N \le 100$). Вторая строка содержит N чисел, представляющих заданный массив. Каждый элемент массива – натуральное число от 1 до 31. Все элементы массива разделены пробелом.

**Выходные данные**

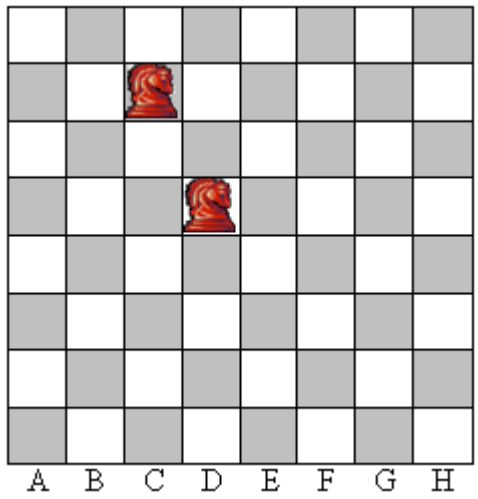
В первую строку выходного файла OUTPUT.TXT нужно вывести числа, которые соответствуют дням месяцев, в которые Вася получил тройки, а во второй строке соответственно расположить числа месяца, в которые Вася получил четверки. В третьей строке нужно вывести «YES», если Вася может рассчитывать на четверку и «NO» в противном случае. При выводе, числа отделяются пробелом.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 5<br>4 16 19 31 2 | 19 31<br>4 16 2<br>YES |
| 2 | 8<br>29 4 7 12 15 17 24 1 | 29 7 15 17 1<br>4 12 24<br>NO |

## 006. Шахматы

*(Время: 1 сек. Память: 16 Мб Сложность: 18%)*




Совсем недавно Вася занялся программированием и решил реализовать собственную программу для игры в шахматы. Но у него возникла проблема определения правильности хода конем, который делает пользователь. Т.е. если пользователь вводит значение «C7-D5», то программа должна определить это как правильный ход, если же введено «E2-E4», то ход неверный. Так же нужно проверить корректность записи ввода: если например, введено «D9-N5», то программа должна определить данную запись как ошибочную. Помогите ему осуществить эту проверку!

**Входные данные**

В единственной строке входного файла INPUT.TXT записан текст хода, который указал пользователь. Пользователь не может ввести строку, длиннее 5 символов.

**Выходные данные**

В выходной файл OUTPUT.TXT нужно вывести «YES», если указанный ход конем верный, если же запись корректна (в смысле правильности записи координат), но ход невозможен, то нужно вывести «NO». Если же координаты не определены или заданы некорректно, то вывести сообщение «ERROR».

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | C7-D5 | YES |
| 2 | E2-E4 | NO |
| 3 | BSN | ERROR |

## 007. Золото племени АББА

*(Время: 1 сек. Память: 16 Мб Сложность: 40%)*

Главный вождь племени Абба не умеет считать. В обмен на одну из его земель вождь другого племени предложил ему выбрать одну из трех куч с золотыми монетами. Но вождю племени Абба хочется получить наибольшее количество золотых монет. Помогите вождю сделать правильный выбор!

**Входные данные**

В единственной строке входного файла INPUT.TXT записаны три натуральных числа через пробел. Каждое из чисел не превышает $10^{100}$.

**Выходные данные**

В выходной файл OUTPUT.TXT нужно вывести одно целое число – максимальное количество монет, которые может взять вождь.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 5 7 3 | 7 |
| 2 | 987531 234 86364 | 987531 |
| 3 | 189285 283 4958439238923098349024 | 4958439238923098349024 |

## 008. Арифметика

*(Время: 1 сек. Память: 16 Мб Сложность: 5%)*

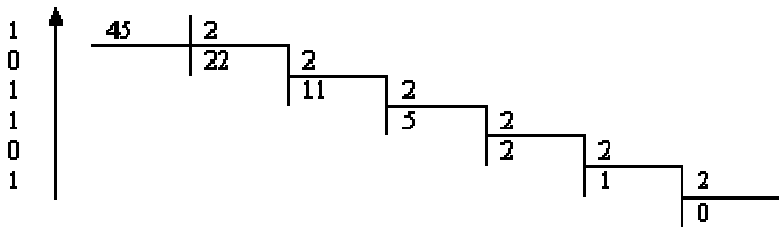
В прошлом году Вася пошел в школу и научился считать. В этом году он изучил таблицу умножения и теперь умеет перемножать любые числа от 1 до 10 без ошибок. Друг Петя рассказал ему про системы счисления, отличные от десятичной. В частности, про двоичную,

восьмеричную и даже шестнадцатеричную. Теперь Вася без труда (но уже с помощью листка и ручки) может перемножать числа от 1 до 10 и в этих системах, используя перевод из нестандартной системы в десятичную и обратно из десятичной. Например, если Васе нужно перемножить числа 101 и 1001 в двоичной системе, то он сначала эти числа переводит в десятичное представление следующим образом:

$(101)_2=1*2^2+0*2^1+1*2^0=4+0+1=5$

$(1001)_2=1*2^3+0*2^2+0*2^1+1*2^0=8+0+0+1=9$

После чего перемножение чисел 5 и 9 Вася с легкостью производит в десятичной системе счисления в уме и получает число 45. Далее производится перевод из десятичной системы счисления в двоичную. Для этого Вася делит число 45 на 2 (порядок системы счисления), запоминая остатки от деления, до тех пор пока в результате не останется число 0:




Ответ составляется из полученных остатков от деления путем их записи в обратном порядке. Таким образом Вася получает результат: $(101)_2 * (1001)_2 = (101101)_2$. Но теперь Вася изучает таблицу умножения чисел от 1 до 100 в десятичной системе счисления, а поскольку запомнить такую таблицу очень сложно, то Васе придется очень долго ее зубрить. Составьте для Васи программу, которая поможет ему проверять свои знания.

**Входные данные**

Во входном файле INPUT.TXT записаны три натуральных числа A, B и C через пробел. Числа A и $B \le 10^2$, а $C \le 10^6$.

**Выходные данные**

В выходной файл нужно вывести YES в том случае, если A*B=C и вывести NO в противном случае.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 8 54 432 | YES |
| 2 | 16 19 777 | NO |

**Разбор**

Эта задача относится к роду тех задач, формулировку которых читать дольше, чем решать саму задачу. Действительно, здесь много лишнего текста: абсолютно ненужной оказывается описание принципов перевода чисел в двоичную систему и обратно (хотя, если Вы с этим еще не знакомы, то это вполне полезно было прочитать). Для понимания того, что же на самом деле здесь нужно сделать достаточно прочтения разделов входных и выходных данных.

Оказывается, что в этой задаче нужно считать три целых числа и проверить: равно ли третье число произведению первых двух. Если да, то вывести «YES» и «NO» иначе. Как видите, все просто.

Несмотря на простоту и в этой задаче бывают ошибки. Например, многие считают, что при выводе ответа регистр не важен и можно выводить вместо «YES» так же «Yes» и «yes». Это неверное мнение, несмотря на то, что в этой системе такое решение будет засчитано. Бывали случаи, когда участники получали 0 баллов вместо 100 за такое решение. Так же при реализации данной программы в среде DOS можно неверно определить тип для хранения переменных. Здесь лучше использовать 4-байтное целое long, а не int. Напомним, что в среде Windows эти типы равнозначны.

## 009. Домашнее задание

*(Время: 1 сек. Память: 16 Мб Сложность: 27%)*

Петя успевает по математике лучше всех в классе, поэтому учитель задал ему сложное домашнее задание, в котором нужно в заданном наборе целых чисел найти сумму всех положительных элементов, затем найти где в заданной последовательности находятся максимальный и минимальный элемент и вычислить произведение чисел, расположенных между ними. Так же известно, что минимальный и максимальный элемент встречаются в заданном множестве чисел только один раз. Поскольку задач такого рода учитель дал Пете около ста, то Петя как сильный программист смог написать программу, которая по заданному набору чисел самостоятельно находит решение. А Вам слабо?

**Входные данные**

В первой строке входного файла INPUT.TXT записано единственное число N – количество элементов массива. Вторая строка содержит N целых чисел, представляющих заданный массив. Все элементы массива разделены пробелом. Каждое из чисел во входном файле не превышает $10^2$ по абсолютной величине.

**Выходные данные**

В единственную строку выходного файла OUTPUT.TXT нужно вывести два числа, разделенных пробелом: сумму положительных элементов и произведение чисел, расположенных между минимальным и максимальным элементами. Значения суммы и произведения не превышают по модулю $3*10^4$.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 5<br>-7 5 -1 3 9 | 17 -15 |
| 2 | 8<br>3 14 -9 4 -5 1 -12 4 | 26 180 |
| 3 | 10<br>-5 1 2 3 4 5 6 7 8 -3 | 36 5040 |

## 010. Уравнение

*(Время: 1 сек. Память: 16 Мб Сложность: 17%)*

Вася в школе изучил квадратные уравнения и понял, как они легко решаются путем вычисления дискриминанта. Но Петя поведал ему о методе решения кубических уравнений вида $A*X^3 + B*X^2 + C*X + D = 0$. На факультативе по математике Васе задали решить около ста уравнений как раз такого вида. Но, к сожалению, Вася забыл формулы, о которых рассказывал ему Петя. Но Васе было известно, что все корни уравнений – целые числа и находятся на отрезке [-100, 100]. Поэтому у Васи есть шанс найти их методом перебора, но для этого ему придется затратить уйму времени, т.к. возможно необходимо будет осуществить перебор нескольких тысяч значений. Помогите Васе написать программу, которая поможет ему найти корни кубических уравнений!

**Входные данные**

В единственной строке входного файла INPUT.TXT записаны 4 числа: A, B, C и D – целые коэффициенты кубического уравнения. Каждый коэффициент по модулю меньше 32768.

**Выходные данные**

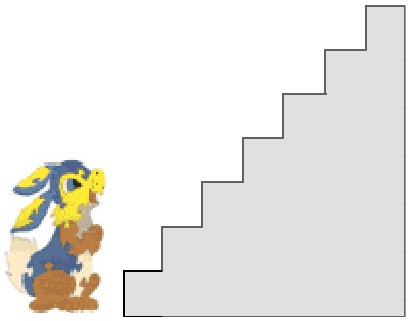
В единственную строку выходного файла OUTPUT.TXT нужно вывести через пробел в порядке возрастания все корни заданного кубического уравнения. Кратные корни следует выводить только один раз.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 1 -3 0 0 | 0 3 |
| 2 | 3 -15 18 0 | 0 2 3 |
| 3 | 1 -7 -33 135 | -5 3 9 |

## 011. Зайчик

*(Время: 1 сек. Память: 16 Мб Сложность: 68%)*



В нашем зоопарке появился заяц. Его поместили в клетку, и чтобы ему не было скучно, директор зоопарка распорядился поставить в его клетке лесенку. Теперь наш зайчик может прыгать по лесенке вверх, перепрыгивая через ступеньки. Лестница имеет определенное количество ступенек N. Заяц может одним прыжком преодолеть не более K ступенек. Для разнообразия зайчик пытается каждый раз найти новый путь к вершине лестницы. Директору любопытно, сколько различных способов есть у зайца добраться до вершины лестницы при заданных значениях K и N. Помогите директору написать программу, которая поможет вычислить это количество. Например, если K=3 и N=4, то существуют следующие маршруты: 1+1+1+1, 1+1+2, 1+2+1, 2+1+1, 2+2, 1+3, 3+1. Т.е. при данных значениях у зайца всего 7 различных маршрутов добраться до вершины лестницы.

**Входные данные**

В единственной строке входного файла INPUT.TXT записаны два натуральных числа K и N ($1 \le K \le N \le 300$). K – максимальное количество ступенек, которое может преодолеть заяц одним прыжком, N – общее число ступенек лестницы.

**Выходные данные**

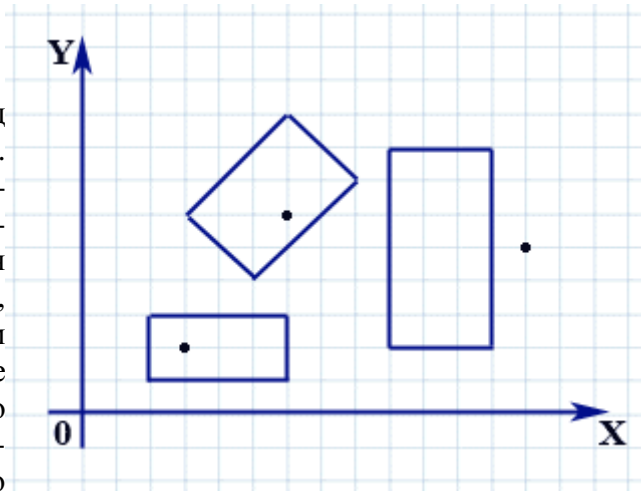
В единственную строку выходного файла OUTPUT.TXT нужно вывести количество возможных вариантов различных маршрутов зайца на верхнюю ступеньку лестницы.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 1 3 | 1 |
| 2 | 2 7 | 21 |
| 3 | 3 10 | 274 |

## 012. Дачники

*(Время: 1 сек. Память: 16 Мб Сложность: 45%)*




Всем известно, что дачники – народ странный, почти такой же, как и программисты. Строят они свои дачи непонятно где, да и выращивают там непонятно что и непонятно зачем. А уж как они туда добираются, это другая история: кто на автобусе, кто на электричке, кто на автомобиле, ну а кто-то вовсе пешком ходит от дома и до самого участка. Так что не стоит удивляться, если вдруг Вы узнаете, что некое садоводческое товарищество располагается на острове, а дачники добираются до него самолетом. Да еще и на этом острове может не быть посадочной полосы, так что высадиться на остров можно, только прыгая с парашютом (мы уж не рассматриваем то, как они возвращаются с дач домой). Рассмотрим этот уникальный случай. Пилот всегда старается осуществить высадку парашютистов таким образом, чтобы дачники приземлялись как можно ближе к своим прямоугольным участкам. Пилоту интересно знать: сколько дачников приземлится на свои участки? Помогите ему решить эту задачу!

**Входные данные**

В первой строке входного файла INPUT.TXT записано натуральное число N ($1 \le N \le 1000$) – количество дачников, далее идут N строк, в каждой из которых описаны координаты каждого дачника и его участка: X Y X1 Y1 X2 Y2 X3 Y3 X4 Y4 где (X,Y) – координаты приземления парашютиста (X1, Y1, X2, Y2, X3, Y3, X4,Y4) – координаты прямоугольного участка на плоскости, указанные последовательно.

Все координаты – целые числа, не превышающие 30000 по абсолютной величине

**Выходные данные**

В выходной файл OUTPUT.TXT нужно вывести количество дачников, приземлившихся на свой участок.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3<br>6 6 3 6 6 9 8 7 5 4<br>13 5 9 2 9 8 12 8 12 2<br>3 2 2 1 2 3 6 3 6 1 | 2 |

## 013. Быки и коровы

*(Время: 1 сек. Память: 16 Мб Сложность: 26%)*

Петя и Вася часто играют в различные логические игры. Недавно Петя поведал Васе о новой игре «Быки и коровы» и теперь они играют в эту игру сутками. Суть игры очень проста: Петя загадывает четырехзначное число, состоящее из различных цифр. Вася отгадывает задуманное Петей число, перебирая возможные варианты. Каждый раз Вася предлагает вариант своего числа, а Петя делает Васе подсказку: сообщает количество быков и коров, после чего Вася с учетом подсказки продолжает отгадывание числа до тех пор, пока не отгадает. Быки – это количество цифр в предложенном Васей числе, совпадающих по значению и стоящих в правильной позиции в задуманном Петей числе. Коровы – количество цифр, совпадающих по значению, но находящихся в неверной позиции. Например, если Петя задумал число 5671, а Вася предложил вариант 7251, то число быков равно 1 (только цифра 1 на своем месте), а число коров равно 2 (только цифры 7 и 5 не на своих местах). Петя силен в математике, но даже он может ошибаться. Помогите Пете написать программу, которая бы по загаданному Петей и предложенному Васей числам сообщала количество быков и коров.

**Входные данные**

В единственной строке входного файла INPUT.TXT записано два четырехзначных натуральных числа A и B через пробел, где A – загаданное Петей число, а B – предложенный Васей вариант.

**Выходные данные**

В выходной файл OUTPUT.TXT нужно вывести два целых числа через пробел – количество быков и коров.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 5671 7251 | 1 2 |
| 2 | 1234 1234 | 4 0 |
| 3 | 2034 6234 | 2 1 |

## 014. НОК

*(Время: 1 сек. Память: 16 Мб Сложность: 24%)*

Требуется написать программу, определяющую наименьшее общее кратное (НОК) чисел a и b.

**Входные данные**

В единственной строке входного файла INPUT.TXT записаны через пробел два натуральных числа – A и B, не превышающие 46340.

**Выходные данные**

В единственную строку выходного файла OUTPUT.TXT нужно вывести одно целое число – НОК чисел A и B.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 36 27 | 108 |
| 2 | 39 65 | 195 |

**Разбор**

*Наименьшее общее кратное* (НОК) двух целых чисел a и b есть наименьшее натуральное число, которое делится на a и b. Обычно обозначается [a, b], а иногда НОК(m, n) или LCM(a, b). Например, НОК(16, 24)=48.

Для нахождения НОК удобно использовать следующее свойство: для любых натуральных чисел a и b верно равенство НОД(a, b)*НОК(a, b)=a*b, откуда получаем, что НОК(a, b)=a*b/НОД(a, b).

В условиях данной задачи можно НОД найти перебором, но более универсально использовать алгоритм Евклида, реализация которого рассмотрена в задаче 148.

## 015. Дороги

*(Время: 1 сек. Память: 16 Мб Сложность: 18%)*

В галактике «Milky Way» на планете «Snowflake» есть N городов, некоторые из которых соединены дорогами. Император галактики «Milky Way» решил провести инвентаризацию дорог на планете «Snowflake». Но, как оказалось, он не силен в математике, поэтому он просит вас сосчитать количество дорог. Требуется написать программу, помогающую императору сосчитать количество дорог на планете «Snowflake».

**Входные данные**

В первой строке входного файла INPUT.TXT записано число N ($0 \le N \le 100$). В следующих N строках записано по N чисел, каждое из которых является единичкой или ноликом. Причем, если в позиции (i, j) квадратной матрицы стоит единичка, то i-ый и j-ый города соединены дорогами, а если нолик, то не соединены.

**Выходные данные**

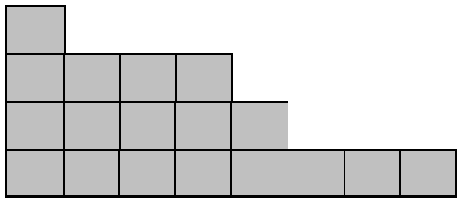
В выходной файл OUTPUT.TXT необходимо вывести число, определяющее количество дорог на планете «Snowflake».

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 5<br>0 1 0 0 0<br>1 0 1 1 0<br>0 1 0 0 0<br>0 1 0 0 0<br>0 0 0 0 0 | 3 |

## 016. Лесенка

*(Время: 1 сек. Память: 16 Мб Сложность: 55%)*



Лесенкой называется набор кубиков, в котором каждый более верхний слой содержит кубиков меньше, чем предыдущий. Требуется написать программу, вычисляющую число лесенок, которое можно построить из N кубиков.

**Входные данные**

Во входном файле INPUT.TXT записано натуральное число N ($1 \le N \le 100$) – количество кубиков в лесенке.

**Выходные данные**

В выходной файл OUTPUT.TXT необходимо вывести число лесенок, которые можно построить из N кубиков.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3 | 2 |
| 2 | 6 | 4 |

## 017. Поле чудес

*(Время: 1 сек. Память: 16 Мб Сложность: 31%)*

Для игры в «Поле чудес» используется круглый барабан, разделенный на сектора, и стрелка. В каждом секторе записано некоторое число. В различных секторах может быть записано одно и то же число. Однажды ведущий игры решил изменить правила. Он сам стал вращать барабан и называть игроку (который барабана не видел) все числа подряд в том порядке, в котором на них указывала стрелка в процессе вращения барабана. Получилось так, что барабан сделал целое число оборотов, то есть последний сектор совпал с первым. После этого, ведущий задал участнику вопрос: какое наименьшее число секторов может быть на барабане? Требуется написать программу, отвечающую на этот вопрос ведущего.

**Входные данные**

В первой строке входного файла INPUT.TXT записано число N – количество чисел, которое назвал ведущий ($2 \le N \le 30000$). Во второй строке записано N чисел, на которые указывала стрелка в процессе вращения барабана. Первое число всегда совпадает с последним (в конце стрелка указывает на тот же сектор, что и в начале). Числа, записанные в секторах барабана – натуральные, не превышающие 32000.

**Выходные данные**

В выходной файл OUTPUT.TXT необходимо вывести одно число – минимальное число секторов, которое может быть на барабане.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 13<br>5 3 1 3 5 2 5 3 1 3 5 2 5 | 6 |
| 2 | 4<br>1 1 1 1 | 1 |
| 3 | 4<br>1 2 3 1 | 3 |

## 018. Факториал

*(Время: 1 сек. Память: 16 Мб Сложность: 42%)*

Требуется вычислить факториал целого числа N. Факториал обозначают как N! и вычисляют по формуле:

N! = 1 * 2 * 3 * … * (N-1) * N, причем 0! = 1.

Так же допустимо рекуррентное соотношение: N! = (N-1)! * N

**Входные данные**

В единственной строке входного файла INPUT.TXT записано одно целое неотрицательное число N (N < 1000).

**Выходные данные**

В выходной файл OUTPUT.TXT нужно вывести одно целое число – значение N!.

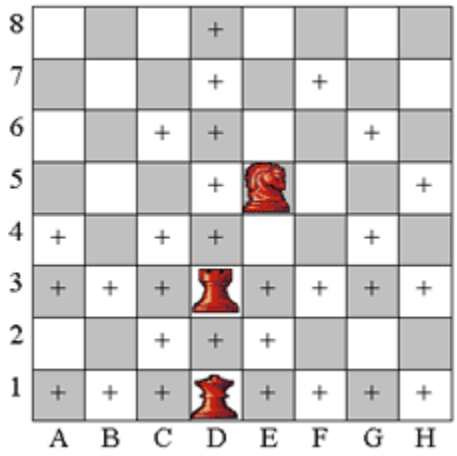
**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 3 | 6 |
| 3 | 5 | 120 |

## 019. Ферзь, ладья и конь

*(Время: 1 сек. Память: 16 Мб Сложность: 34%)*

На шахматной доске 8x8 расположены три фигуры: ферзь, ладья и конь. Требуется определить количество пустых полей доски, которые находятся под боем. Для простоты будем полагать, что фигуры могут «бить» через другие фигуры. Например, в рассмотренной справа ситуации будем считать, что ферзь бьет D5 через ладью.




**Входные данные**

В единственной строке входного файла INPUT.TXT записаны через пробел координаты расположения трех фигур: ферзя, ладьи и коня соответственно. Каждая координата состоит из одного латинского символа (от A до H) и одной цифры (от 1 до 8).

**Выходные данные**

В выходной файл OUTPUT.TXT нужно вывести количество пустых полей, которые бьют указанные во входных данных фигуры.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | D1 D3 E5 | 29 |

## 020. Пилообразная последовательность

*(Время: 1 сек. Память: 16 Мб Сложность: 38%)*

Последовательность $a_1, a_2, a_3, \ldots, a_{n-1}, a_n$ называется пилообразной, если она удовлетворяет одному из следующих условий:

1) $a_1 < a_2 > a_3 < \ldots > a_{n-1} < a_n$,

2) $a_1 > a_2 < a_3 > \ldots < a_{n-1} > a_n$.

Дана числовая последовательность. Требуется определить длину самой длиной ее пилообразной непрерывной подпоследовательности.

**Входные данные**

В первой строке входного файла INPUT.TXT записано натуральное число N – количество элементов последовательности. Во второй строке файла через пробел записаны N элементов целочисленной последовательности $\{a_i\}$. Ограничения: $N<10^6$, $|a_i| < 32000$.

**Выходные данные**

В единственную строку выходного файла OUTPUT.TXT нужно вывести одно целое число – длину самой длиной непрерывной пилообразной подпоследовательности.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3<br>1 2 3 | 2 |
| 2 | 12<br>5 7 6 3 4 2 7 1 8 9 4 5 | 7 |
| 3 | 5<br>1 -2 3 -4 5 | 5 |

## 021. Зарплата

*(Время: 1 сек. Память: 16 Мб Сложность: 4%)*

В отделе работают 3 сотрудника, которые получают заработную плату в рублях. Требуется определить: на сколько зарплата самого высокооплачиваемого из них отличается от самого низкооплачиваемого.

**Входные данные**

В единственной строке входного файла INPUT.TXT записаны размеры зарплат всех сотрудников через пробел. Каждая заработная плата – это натуральное число, не превышающее $10^5$.

**Выходные данные**

В выходной файл OUTPUT.TXT необходимо вывести одно целое число – разницу между максимальной и минимальной зарплатой.

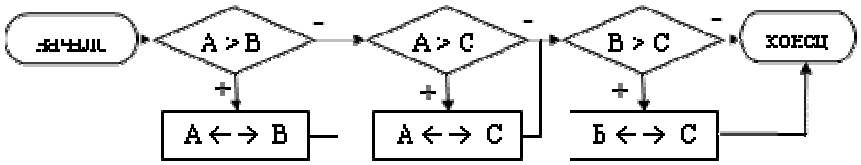
**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 100 500 1000 | 900 |
| 2 | 36 11 20 | 25 |

**Разбор**

В данной задаче необходимо найти наибольшее и наименьшее значение и вывести их разность. Для этого проще всего упорядочить заданные числа A, B и C в порядке неубывания ($A \le B \le C$) и тогда значение C-A будет решением этой задачи. Для этого можно было бы прибегнуть к принципу чайника и отсортировать массив из трех элементов методом «пузырька», например. Но это решение не самое разумное в данном случае.

Здесь мы можем применить тот же метод «пузырька», но без циклов и массивов. Действительно, за 3 сравнения можно достичь желаемого результата. Сначала сравнивая A и B мы можем в A поместить наименьшее из них, поменяв их местами. Далее сравнивая A и C мы поместим в A наименьший из 3х чисел элемент. A после сравнения B и C в C получим наибольший. Описанный выше алгоритм можно представить в виде следующей блок-схемы:




На всякий случай напомним как с помощью третьей переменной можно поменять значения переменных местами:

```
x=a; a=b; b=x;
```

В языке Си для различных целочисленных переменных a и b можно использовать более красивую и короткую запись без использования третьей переменной:

```
a^=b^=a^=b;
```

## 022. Единицы

*(Время: 1 сек. Память: 16 Мб Сложность: 16%)*

На уроках информатики вас, наверное, учили переводить числа из одних систем счисления в другие и выполнять другие подобные операции. Пришло время продемонстрировать эти знания. Найдите количество единиц в двоичной записи заданного числа.

**Входные данные**

Во входном файле INPUT.TXT записано целое число n ($0 \le n \le 2*10^9$).

**Выходные данные**

В единственную строку выходного файла OUTPUT.TXT нужно вывести одно целое число – количество двоичных единиц в записи числа n.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 5 | 2 |
| 2 | 7 | 3 |

**Разбор**

Это достаточно простая задача, т.к. ответом является сумма цифр заданного числа в двоичном представлении. Реализация алгоритма решения может иметь следующий вид:

```
read(n);
s = 0;
while(n>0){
  s = s + n mod 2;
  n = n div 2;
}
write(s);
```

## 023. Гадание

*(Время: 1 сек. Память: 16 Мб Сложность: 13%)*

Как и многие другие девочки, Маша любит разные гадания. Некоторое время назад Маша узнала новый способ гадать на числах – для какого-нибудь интересующего ее натурального числа n надо посчитать сумму всех чисел, на которые n делится без остатка. Маша не очень любит арифметику, и попросила вас написать программу, которая автоматизирует процесс гадания.

**Входные данные**

В единственной строке входного файла INPUT.TXT записано натуральное число n ($0 \le n \le 1000$), которое Маша была вынуждена сообщить.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите сумму всех натуральных делителей числа n.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 6 | 12 |
| 2 | 10 | 18 |

## 024. Вырубка деревьев

*(Время: 1 сек. Память: 16 Мб Сложность: 46%)*

Король Флатландии решил вырубить некоторые деревья, растущие перед его дворцом. Деревья перед дворцом короля посажены в ряд, всего там растет n деревьев, расстояния между соседними деревьями одинаковы.

После вырубки перед дворцом должно остаться m деревьев, и расстояния между соседними деревьями должны быть одинаковыми. Помогите королю выяснить, сколько существует способов вырубки деревьев.

Требуется написать программу, которая по заданным числам n и m определит, сколько существует способов вырубки некоторых из n деревьев так, чтобы после вырубки осталось m деревьев и соседние деревья находились на равном расстоянии друг от друга.

**Входные данные**

Входной файл INPUT.TXT содержит два целых числа n и m ($0 \le m , n \le 1000$).

**Выходные данные**

В единственную строку выходного файла OUTPUT.TXT нужно вывести одно целое число – искомое число способов.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 5 3 | 4 |

**Пояснение к примеру**

Если обозначить условно исходное расположение деревьев перед дворцом как «TTTTT», то возможные результаты после вырубки следующие: «TTT..», «.TTT.», «..TTT», «T.T.T».

## 025. Больше-меньше

*(Время: 1 сек. Память: 16 Мб Сложность: 3%)*

Одна из основных операций с числами – их сравнение. Мы подозреваем, что вы в совершенстве владеете этой операцией и можете сравнивать любые числа, в том числе и целые. В данной задаче необходимо сравнить два целых числа.

**Входные данные**

В двух строчках входного файла INPUT.TXT записаны числа A и B, не превосходящие по абсолютной величине $2*10^9$.

**Выходные данные**

Запишите в выходной файл один символ "<", если A < B, ">", если A > B и "=", если A=B.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 5<br>7 | < |
| 2 | -7<br>-12 | > |
| 3 | 13<br>13 | = |

**Разбор**

В этой задаче нужно считать два числа и произвести их сравнение с помощью условного оператора if, можно даже без else. Заметим, что при чтении целых чисел можно не обращать внимание на то, что они записаны в разных строках: при использовании записи scanf("%d%d", &x, &y) или read(x,y) числа будут считаны так же успешно, как если бы они были записаны через пробел.

## 026. Две окружности

*(Время: 1 сек. Память: 16 Мб Сложность: 17%)*

На плоскости даны две окружности. Требуется проверить, пересекаются ли они.

**Входные данные**

Входной файл INPUT.TXT состоит из двух строк. На каждой строке записана информация об одной окружности – координаты ее центра x и y (целые числа, по модулю не превосходящие 5000) и радиус (целое число $1 \le r \le 1000$).

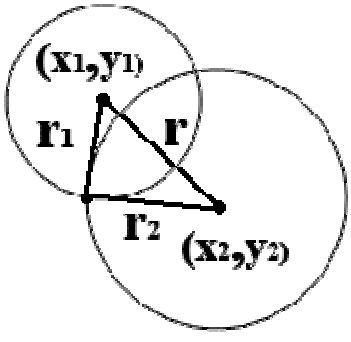
**Выходные данные**

В выходной файл OUTPUT.TXT выведите «YES», если окружности пересекаются, и «NO» в противном случае.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 0 0 2<br>0 3 2 | YES |
| 2 | 1 1 1<br>4 4 1 | NO |

**Разбор**




Для начала определимся с тем, что нам известны радиусы окружностей и они равны r1 и r2. Так же по формуле расстояния между точками мы можем вычислить расстояние между центрами данных окружностей:

$r = sqrt( (x2-x1)^2 + (y2-y1)^2 )$

Заметим так же, что окружности будут пересекаться тогда и только тогда, когда возможен треугольник со сторонами r1, r2 и r. Фигуру, две стороны которой лежат на третьей или одна из сторон имеет нулевую длину так же будем считать треугольником, т.к. окружности могут друг друга касаться(r=r1+r2), либо полностью совпадать (r=0).

Треугольник считается возможным если сумма двух любых его сторон не меньше третьей. Т.е. в нашем случае достаточно проверить, что r1+r2 ≥ r и r+r2 ≥ r1 и r+r1 ≥ r2. При этом желательно использовать вещественные типы данных. Так же можно провести аккуратное сравнение с учетом возможных погрешностей при вычислениях.

## 027. Художник

*(Время: 1 сек. Память: 16 Мб Сложность: 31%)*

Известный художник решил написать новый шедевр. После многих дней усердной работы он захотел исследовать свое творение. Художник вспомнил, что картина писалась следующим образом: сначала был взят белый холст, имеющий форму прямоугольника шириной w и высотой h. Затем художник нарисовал на этом холсте n прямоугольников со сторонами, параллельными сторонам холста и вершинами, расположенными в целочисленных координатах. Помогите художнику определить площадь незакрашенной части холста.

**Входные данные**

Первая строка входного файла INPUT.TXT содержит два натуральных числа w и h ($1 \le w, h \le 100$). Во второй строке записано целое число n ($1 \le n \le 5000$) – количество прямоугольников. Следующие n строк содержат информацию о всех прямоугольниках. Каждая строка описывает один прямоугольник в виде четырех чисел x1, y1, x2, y2 , где (x1, y1) и (x2, y2) – координаты левого верхнего и правого нижнего угла прямоугольника соответственно.

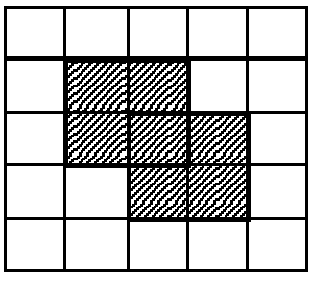
**Выходные данные**

Выведите в выходной файл OUTPUT.TXT одно целое число – площадь незакрашенной части холста.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 5 5<br>2<br>1 1 3 3<br>2 2 4 4 | 18 |
| 2 | 6 7<br>3<br>0 0 5 5<br>1 1 4 4<br>2 2 3 3 | 17 |

## Разбор




В этой задаче необходимо понять, что координаты закрашиваемых прямоугольников определяются не координатами ячеек, а координатами сетки, т.е. левая верхняя координата имеет значение (0, 0) в то время как правая нижняя - (w, h). Это следует из примеров, первый из которых представлен на рисунке справа, где видно, что действительно 18 клеток таблицы не закрашены. В этой задаче необходимо понять, что координаты закрашиваемых прямоугольников определяются не координатами ячеек, а координатами сетки, т.е. левая верхняя координата имеет значение (0, 0) в то время как правая нижняя – (w, h). Это следует из примеров, первый из которых представлен на рисунке справа, где видно, что в этой задаче необходимо понять, что координаты закрашиваемых прямоугольников определяются не координатами ячеек, а координатами сетки, т.е. левая верхняя координата имеет значение (0, 0) в то время как правая нижняя – (w, h). Это следует из примеров, первый из которых представлен на рисунке справа, где видно, что действительно 18 клеток таблицы не закрашены.

Данная задача решается «в лоб». Можно определить матрицу a[1..n][1..m] и пошагово считывать координаты противоположных вершин прямоугольника и сразу же заполнять единицами этот прямоугольник в матрице. Предварительно матрицу необходимо обнулить. По завершении процесса можно подсчитать число оставшихся нулей в матрице, это и будет ответом на задачу. Максимальное число простых операций может быть равно 50 000 000. Несмотря на такое, казалось бы огромное число решение задачи укладывается в 1 секунду. Дело в том, что проводимые операции – это заполнение элементов массива числами, которые выполняются очень быстро. Если бы столько же раз нам нужно было выполнить серию умножений, то мы вряд ли смогли бы рассчитывать на успех.

Приведем пример решения данной задачи на алгоритмическом языке:

```
int a[1..100][1..100]={0...0};

read(w,h,n);
for i=1..n{
  read(x1,y1,x2,y2);
  for y=y1+1..y2{
    for x=x1+1..x2{
      a[y][x]=1;
    }
  }
}
c=0;
for y=1..h{
  for x=1..w{
    c=c+1-a[y][x];
  }
}
write(c);
```

## 028. Симметрия

*(Время: 1 сек. Память: 16 Мб Сложность: 19%)*

Многие из вас, вероятно, знакомы с понятием симметрии относительно прямой. Пусть на плоскости расположена прямая L и точка A. Точка B называется симметричной точке A относительно прямой L, если отрезок AB перпендикулярен прямой L и делится пополам точкой пересечения с ней. В частности, если точка A лежит на прямой L, то точка B совпадает с точкой A.

Задана прямая L, параллельная одной из осей координат, и точка A. Найдите точку B, симметричную A относительно L.

**Входные данные**

Первая строка входного файла INPUT.TXT содержит 4 числа: $x_1$, $y_1$, $x_2$, $y_2$ – координаты двух различных точек, через которые проходит прямая L. Вторая строка входного файла содержит 2 числа $x_A$ и $y_A$ – координаты точки A. Все числа во входном файле целые и не превосходят $10^8$ по модулю.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите числа $x_B$ и $y_B$ – координаты точки B.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 0 0 0 1<br>10 10 | -10 10 |
| 2 | 0 0 1 0<br>10 10 | 10 -10 |

## 029. Компьютерная игра

*(Время: 1 сек. Память: 16 Мб Сложность: 38%)*

Вы можете вспомнить хоть одного своего знакомого до двадцатилетнего возраста, который в детстве не играл в компьютерные игры? Если да, то может быть вы и сами не знакомы с этим развлечением? Впрочем, трудностей при решении этой задачи это создать не должно.

Во многих старых играх с двумерной графикой можно столкнуться с подобной ситуацией. Какой-нибудь герой прыгает по платформам (или островкам), которые висят в воздухе. Он должен перебраться от одного края экрана до другого. При этом при прыжке с одной платформы на соседнюю, у героя уходит |y2-y1| единиц энергии, где y1 и y2 – высоты, на которых расположены эти платформы. Кроме того, у героя есть суперприем, который позволяет перескочить через платформу, но на это затрачивается 3*|y3-y1| единиц энергии. Конечно же, энергию следует расходовать максимально экономно.

Предположим, что вам известны координаты всех платформ в порядке от левого края до правого. Сможете ли вы найти, какое минимальное количество энергии потребуется герою, чтобы добраться с первой платформы до последней?

**Входные данные**

В первой строке входного файла INPUT.TXT записано количество платформ n ($1 \le n \le 30000$). Вторая строка содержит n натуральных чисел, не превосходящих 30000 – высоты, на которых располагаются платформы.

**Выходные данные**

В выходной файл OUTPUT.TXT запишите единственное число – минимальное количество энергии, которую должен потратить игрок на преодоление платформ (конечно же, в предположении, что cheat-коды использовать нельзя).

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3<br>1 5 10 | 9 |
| 2 | 3<br>1 5 2 | 3 |

## 030. Часы

*(Время: 1 сек. Память: 16 Мб Сложность: 35%)*

Петя очень любит наблюдать за электронными часами. Он целыми днями смотрел на часы и считал, сколько раз встречается каждая цифра. Через несколько месяцев он научился по любому промежутку времени говорить, сколько раз на часах за это время встретится каждая цифра, и очень гордился этим.

Вася решил проверить Петю, но он не знает, как это сделать. Вася попросил Вас помочь ему. Напишите программу, решающую эту задачу.

**Входные данные**

Первая и вторая строки входного файла INPUT.TXT содержат начало и конец промежутка времени соответственно. Начальное время не превосходит конечное. Время задается в формате hh:mm:ss ($0 \le hh < 24$, $0 \le mm < 60$, $0 \le ss < 60$). hh, mm, ss дополнены ведущими нулями до двух символов. Эти нули также учитываются при подсчете числа цифр.

**Выходные данные**

Выходной файл OUTPUT.TXT должен содержать 10 строк. В i-ой строке должно быть написано, сколько раз встречается цифра i-1.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 23:59:58<br>23:59:59 | 0<br>0<br>2<br>2<br>0<br>4<br>0<br>0<br>1<br>3 |
| 2 | 13:24:09<br>13:24:40 | 5<br>45<br>45<br>45<br>36<br>3<br>3<br>3<br>3<br>4 |

## 031. Неподвижные точки

*(Время: 1 сек. Память: 16 Мб Сложность: 57%)*

Перестановкой P[1..n] размера n называется набор чисел от 1 до n, расположенных в определенном порядке. При этом в нем должно присутствовать ровно один раз каждое из этих чисел. Примером перестановок являются 1, 3, 4, 5, 2 (для n=5) и 3, 2, 1 (для n=3), а, например, 1, 2, 3, 4, 5, 1 перестановкой не является, так как число 1 встречается два раза.

Число i называется неподвижной точкой для перестановки P, если P[i] = i. Например, в перестановке 1, 3, 4, 2, 5 ровно две неподвижных точки: 1 и 5, а перестановка 4, 3, 2, 1 не имеет неподвижных точек.

Даны два числа: n и k. Найдите количество перестановок размера n с ровно k неподвижными точками.

**Входные данные**

Входной файл INPUT.TXT содержит два целых числа n ($1 \le n \le 9$) и k ($0 \le k \le n$).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите ответ на задачу.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 5 2 | 20 |
| 2 | 9 6 | 168 |
| 3 | 2 1 | 0 |
| 4 | 9 0 | 133496 |

## 032. Годовой баланс

*(Время: 1 сек. Память: 16 Мб Сложность: 33%)*

В конторе «Рога и Копыта» подходит время подведения годового баланса. В бухгалтерию поступили сведения о том, что, согласно документам, суммарный расход составил a рублей, а суммарный приход – b рублей. Поскольку с реальным положением дел эти цифры все равно не имеют ничего общего, бухгалтер решил реализовать следующую свою идею.

Как известно, при наборе чисел на компьютере люди часто вводят цифры в неправильном порядке. Поэтому бухгалтер хочет найти такой способ переставить цифры в числах a и b, чтобы в результате разность a-b (и, соответственно, количество денег, которые он положит к себе в карман), была максимальна, а в случае можно будет сослаться на ошибку секретаря. При этом нельзя забывать о знаке чисел и о том, что ноль не может быть первой цифрой числа. Напишите программу, которая поможет бухгалтеру.

**Входные данные**

Входной файл INPUT.TXT содержит два целых числа a и b ($-10^9 < a, b < 10^9$).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите одно целое число – наибольшую разность чисел, первое из которых может быть получено перестановкой цифр a, а второе – перестановкой цифр b.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 18<br>10 | 71 |
| 2 | 1<br>-23 | 33 |

**Разбор**

Для того чтобы значение **a-b** было максимально, необходимо из **a** получить максимально возможное число, а из **b**, наоборот, определить наименьшее. Если число неотрицательно, то для получения минимума цифры сортируются в порядке неубывания, а при вычислении максимума, соответственно, в порядке невозрастания. Для отрицательных чисел ситуация с сортировкой цифр меняется с точностью до наоборот. Причем, нужно заметить, что при сортировке цифр по возрастанию нужно избежать момента, где пришлось бы поставить **0** на первое место (этого можно достичь, например, при сортировке пузырьком, если при сравнении первых двух соседних элементов не менять их в том случае, когда второй – это ноль).

Сортировку цифр в числе проще проводить, если преобразовать само число в строку, и работать с обычным массивом символов. Причем, знак «-» не обязательно включать в этот массив. Отсортировав его, например пузырьком, можно сделать обратное преобразование строки в число. После нахождения максимального **a** и минимального **b** останется просто вывести значение **a-b**.

Для хранения чисел можно использовать обычный 4-байтовый целый тип (long или longint например), т.к. результат вычислений не может превзойти значения 1999999998 по абсолютной величине. Если бы ограничения на **a** и **b** были не до миллиарда, а до двух миллиардов, то итоговая разница могла бы достигать значения, большего 19 миллиардов, где пришлось бы использовать такие 8-байтоые целые типы как int64 или __int64.

## 033. Два бандита

*(Время: 1 сек. Память: 16 Мб Сложность: 4%)*

Бандиты Гарри и Ларри отдыхали на природе. Решив пострелять, они выставили на бревно несколько банок из-под пива (не больше 10). Гарри начал простреливать банки по порядку, начиная с самой левой, Ларри – с самой правой. В какой-то момент получилось так, что они одновременно прострелили одну и ту же последнюю банку.

Гарри возмутился и сказал, что Ларри должен ему кучу денег за то, что тот лишил его удовольствия прострелить несколько банок. В ответ Ларри сказал, что Гарри должен ему еще больше денег по тем же причинам. Они стали спорить кто кому сколько должен, но никто из них не помнил сколько банок было в начале, а искать простреленные банки по всей округе было неохота. Каждый из них помнили только, сколько банок прострелил он сам.

Определите по этим данным, сколько банок не прострелил Гарри и сколько банок не прострелил Ларри.

**Входные данные**

В единственной строке входного файла INPUT.TXT записано 2 числа – количество банок, простреленных Гарри и Ларри соответственно.

**Выходные данные**

В файл OUTPUT.TXT выведите 2 числа – количество банок, не простреленных Гарри и Ларри соответственно.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 4 7 | 6 3 |

**Разбор**

Прочитав эту задачу, возникает желание узнать: сколько всего было банок? На самом деле, это вычисляется достаточно просто: c=a+b-1, где единицу мы отняли, т.к. одна из банок простреляна дважды, поэтому количество банок равно числу выстрелов (все банки ведь простреляны) без единицы. Поэтому Гарри не простреляли c-a банок, а Ларри c-b банок.

Оказывается, что можно обойтись и двумя переменными без вычисления общего количества банок. Действительно, каждый бандит не простреляли именно те банки, которые простреляли другой, кроме единственной, в которую они выстрелили вместе. Таким образом, Гарри не простреляли b-1 банок, а Ларри – a-1 банок.

Удачи в чтении a и b, а так же выводе b-1 и a-1!

## 034. Секретное сообщение

*(Время: 1 сек. Память: 16 Мб Сложность: 46%)*

На секретную базу в Арктике поступила шифровка – последовательность из n десятичных цифр. Она содержит номер секретной базы в Антарктиде, который является последовательностью из k десятичных цифр. При этом для того, чтобы отличить его от ненужной Вам информации, он повторен в шифровке хотя бы два раза (возможно, эти два вхождения перекрываются).

Напишите программу, которая по шифровке и длине номера секретной базы определяет, содержит ли шифровка номер базы. Учтите, что у базы может быть несколько номеров, и все они могут быть переданы в шифровке.

**Входные данные**

Первая строка входного файла INPUT.TXT содержит два целых числа: n ($1 \le n \le 10^5$) и k ($1 \le k \le 5$) – длину шифровки и длину номера секретной базы соответственно. Вторая строка содержит n цифр – шифровку. Помните, что цифры в шифровке не разделяются пробелами.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите «YES», если шифровка содержит номер секретной базы, и «NO» в противном случае.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 10 5<br>0123456789 | NO |
| 2 | 13 2<br>0123400056789 | YES |

## 035. Конечные автоматы

*(Время: 1 сек. Память: 16 Мб Сложность: 11%)*

Однажды известный профессор обнаружил описания k конечных автоматов. По его мнению, нетривиальность конечного автомата, имеющего n состояний и m переходов, можно описать целым числом d = 19m + (n + 239)*(n + 366) / 2 . Чем больше d, тем больший интерес для науки представляет изучение его свойств.

Помогите профессору вычислить нетривиальность имеющихся у него автоматов.

**Входные данные**

Первая строка входного файла INPUT.TXT содержит целое число k ($1 \le k \le 10000$) – количество конечных автоматов. Следующие k строк содержат по два целых числа ni ($1 \le n_i \le 1000$) и mi ($1 \le m_i \le 26n_i^2$) – число состояний и число переходов i-го автомата.

**Выходные данные**

Выходной файл OUTPUT.TXT должен состоять из k строк. На i-й строке выходного файла выведите одно число – нетривиальность i-го автомата.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 4<br>2 0<br>13 20<br>5 23<br>18 6 | 44344<br>48134<br>45699<br>49458 |
| 2 | 2<br>15 20<br>1000 26000 | 48767<br>1340237 |

## 036. Постулат Бертрана

*(Время: 1 сек. Память: 16 Мб Сложность: 38%)*

Постулат Бертрана (теорема Бертрана-Чебышева, теорема Чебышева) гласит, что для любого $n > 1$ найдется простое число p в интервале $n < p < 2n$. Такая гипотеза была выдвинута в 1845 году французским математиком Джозефем Бертраном (проверившим ее до n=3000000) и доказана в 1850 году Пафнутием Чебышевым. Раманужан в 1920 году нашел более простое доказательство, а Эрдеш в 1923 – еще более простое.

Ваша задача состоит в том, чтобы решить несколько более общую задачу – а именно по числу n найти количество простых чисел p из интервала $n < p < 2n$.

Напомним, что число называется простым, если оно делится только само на себя и на единицу.

**Входные данные**

Входной файл INPUT.TXT содержит целое число n ($2 \le n \le 50000$).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите одно число – ответ на задачу.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 2 | 1 |
| 2 | 239 | 39 |
| 3 | 3000 | 353 |

**Разбор**

Данная задача сводится к поиску простых чисел, меньших 2*n и подсчету тех из них, которые попадают в интервал (n, 2*n). Быстрый алгоритм поиска простых чисел рассмотрен при разборе задачи 349 «Простые числа». Стоит полагать, что Вы сможете с легкостью его доработать до алгоритма, решающего настоящую задачу.

## 037. Сжимающий оператор

*(Время: 1 сек. Память: 16 Мб Сложность: 34%)*

Оператором A, действующим из множества X в множество Y (или просто оператором из X в Y) называется правило, согласно которому каждому элементу x множества X сопоставляется элемент y=Ax из множества Y. Пусть X и Y – множества точек на плоскости. Оператор A из X в Y называется сжимающим с коэффициентом q, где q – вещественное число из

полуинтервала [0, 1), если для любого x из X выполнено A||x|| ≤ q*||x|| (здесь ||x|| - норма точки x – расстояние от x до начала координат). Проще говоря, оператор называется сжимающим с коэффициентом q если он сопоставляет каждой точке точку, которая не менее, чем в q раз ближе к началу координат.

Для заданного оператора A требуется проверить является ли он сжимающим с коэффициентом q.

**Входные данные**

Первая строка входного файла INPUT.TXT содержит количество точек n ($1 \le n \le 100$) и число q ($0 \le q < 1$), заданное не более чем с 3 знаками после десятичной точки. Следующие n строк содержат по 4 целых числа, по модулю не превосходящих 1000, разделенные пробелами – координаты точки множества X и сопоставленной ей точки из множества Y.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите одно слово: “Yes” если оператор является сжимающим с коэффициентом q и “No” в противном случае.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 2 0.5<br>0 10 5 0<br>10 0 0 1 | Yes |
| 2 | 2 0.1<br>0 10 5 0<br>10 0 0 1 | No |
| 3 | 2 0.9<br>0 0 0 0<br>10 0 0 1 | Yes |

## 038. Игра - 2

*(Время: 1 сек. Память: 16 Мб Сложность: 60%)*

Вы любите играть в игры? Конечно, любите! Но про эту игру, возможно, ничего не знаете и не слышали даже. Что ж, расскажем о новой игре. На доске написана последовательность n целых чисел. Играют двое. На очередном ходе игрок выбирает число с правого или с левого края последовательности, затем это число стирается и последовательность становится на одно число меньше, а ход переходит к противнику. Выигрывает тот, кто наберет в сумме больше. Написать программу, определяющую победителя в конкретной игре, при условии, что игроки будут играть оптимально.

**Входные данные**

В первой строке входного файла INPUT.TXT записано целое число n ($0 < n < 100$). Во второй строке через пробел заданы n натуральных чисел, не превосходящих 1000.

**Выходные данные**

В единственную строку выходного файла OUTPUT.TXT нужно вывести 1, если победит первый игрок, 2 – если победит второй игрок и 0 – в случае ничьей.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 4<br>3 2 5 4 | 1 |
| 2 | 6<br>5 5 5 5 5 5 | 0 |
| 3 | 9<br>2 1 3 2 9 1 2 3 1 | 2 |
| 4 | 10<br>2 5 3 12 4 6 13 7 1 3 | 1 |

## 039. Волосатый бизнес

*(Время: 1 сек. Память: 16 Мб Сложность: 32%)*

Одного неформала выгнали с работы, и теперь ему надо как-то зарабатывать себе на пиво и сигареты. Поразмыслив, он решил, что сможет иметь очень неплохие деньги на продаже собственных волос. Известно, что пункты приема покупают волосы произвольной длины стоимостью C у.е. за каждый сантиметр. Так как волосяной рынок является очень динамичным, то цена одного сантиметра волос меняется каждый день как и курс валют. Неформал является очень хорошим бизнес-аналитиком. Он смог вычислить, какой будет цена одного сантиметра волос в каждый из ближайших N дней (для удобства пронумеруем дни в хронологическом порядке от 0 до N-1). Теперь он хочет определить, в какие из этих дней ему следует продавать волосы, чтобы по истечению всех N дней заработать максимальное количество денег. Заметим, что волосы у неформала растут только ночью и вырастают на 1 сантиметр за ночь. Следует также учесть, что до 0-го дня неформал с горя подстригся наголо и к 0-му дню длина его волос составляла 1 сантиметр.

**Входные данные**

В первой строке входного файла INPUT.TXT записано целое число N ($0 < N \le 100$). Во второй строке через пробел заданы N натуральных чисел, не превосходящих 100, соответствующие стоимости C[i] 1 сантиметра волос за каждый i-й день.

**Выходные данные**

В единственную строку выходного файла OUTPUT.TXT нужно вывести максимальную денежную сумму, которую может заработать неформал за N дней.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 5<br>73 31 96 24 46 | 380 |
| 2 | 10<br>1 2 3 4 5 6 7 8 9 10 | 100 |
| 3 | 10<br>10 9 8 7 6 5 4 3 2 1 | 55 |

## Разбор

В этой задаче нужно понять в какой день на текущий момент времени выгоднее всего сдавать волосы. Все просто: сдавать их нужно в тот день, когда цена максимальна среди оставшихся дней. Т.е. пока еще общий срок истечения дней не окончен каждый раз нужно находить максимальный элемент массива C[k] (k=p+1..n-1) и сдавая все волосы получать свои C[k]*(k-p+2) у.е., где p – номер последнего для сдачи волос. За первый день сдачи можно считать p=-1.

## 040. 2^N

*(Время: 1 сек. Память: 16 Мб Сложность: 30%)*

Необходимо вычислить значение $2^n$.

**Входные данные**

В единственной строке входного файла INPUT.TXT записано натуральное число n ($0 < n < 1000$).

**Выходные данные**

В единственную строку выходного файла OUTPUT.TXT нужно вывести значение $2^n$.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3 | 8 |
| 2 | 10 | 1024 |
| 3 | 72 | 4722366482869645213696 |

# 041. Цифровая сортировка

*(Время: 1 сек. Память: 16 Мб Сложность: 29%)*

На планете «Аурон» атмосфера практически отсутствует, поэтому она известна своими перепадами температур в различных точках. Известно, что эти перепады колеблются от -100 до 100 градусов. Нашим специалистам удалось выяснить значения температур в N точках этой планеты. К сожалению, эти значения вычислены с большими погрешностями, поэтому их решили округлить до целых чисел. Хотелось бы наглядно видеть участки с повышенной и пониженной температурой. Вам требуется помочь. Вы должны упорядочить температуры участков по неубыванию.

**Входные данные**

В первой строке входного файла INPUT.TXT задано натуральное число N – количество участков ($N \le 10^6$). Во второй строке через пробел записаны целые значения температур этих участков, не превосходящие 100 по абсолютной величине.

**Выходные данные**

В единственную строку выходного файла OUTPUT.TXT нужно вывести разделенные пробелом значения температур всех известных участков, которые должны следовать друг за другом в порядке неубывания.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3<br>9 -20 14 | -20 9 14 |
| 2 | 10<br>12 7 92 5 18 4 32 48 11 74 | 4 5 7 11 12 18 32 48 74 92 |

**Разбор**

В этой задаче нужно отсортировать целочисленный массив, но сложность ее в том, что массив может содержать очень большое количество элементов (до миллиона). Здесь может не пройти даже алгоритм быстрой сортировки, но несмотря на это задачу можно решить, применяя так называемую цифровую сортировку (DigitalSort).

Дело в том, что диапазон возможных значений элементов массива ограничен и поэтому вовсе не обязательно хранить весь массив в памяти, достаточно считывая его поэлементно вычислять для каждого возможного элемента x количество таких элементов в массиве. Для этого определяется некоторый массив d[x], в котором каждый элемент – это счетчик количества элементов исходного массива a[i], т.е в итоге в d[x] мы предполагаем получить количество элементов массива a[i], которые равны x. Достичь этого можно следующим образом: при чтении очередного элемента a[i] увеличивать значение d[a[i]] на единицу. По завершении этого процесса достаточно вывести d[x] раз в порядке увеличения x (по всему возможному диапазону) значение x. Данный алгоритм имеет порядок сложности O(N).

Решение данной задачи на алгоритмическом языке:

```
int d[-100..100]={0...0};

read(n);

for i=1..n{
  read(x);
  d[x]=d[x]+1;
}

for x=-100..100{
  for i=1..d[x]{
    write(x, ' ');
  }
}
```

Данный алгоритм не является универсальным и применим исключительно к массивам с ограниченным диапазоном возможных значений.

## 042. Драконы

*(Время: 1 сек. Память: 16 Мб Сложность: 40%)*

Известно, что у дракона может быть несколько голов и его сила определяется числом голов. Но как определить силу драконьей стаи, в которой несколько драконов и у каждого из них определенное число голов? Вероятно, вы считаете, что это значение вычисляется как сумма всех голов? Это далеко не так, иначе было бы слишком просто вычислить силу драконьей стаи. Оказывается, что искомое значение равно произведению значений числа голов каждого из драконов. Например, если в стае 3 дракона, у которых 3, 4 и 5 голов соответственно, то сила равна 3*4*5 = 60. Предположим, что нам известно суммарное значение голов драконьей стаи, как нам вычислить максимально возможное значение силы этого логова драконов? Именно эту задачу Вам и предстоит решить.

**Входные данные**

В единственной строке входного файла INPUT.TXT записано натуральное число N (0 < N < 100) – количество голов драконьей стаи.

**Выходные данные**

В единственную строку выходного файла OUTPUT.TXT нужно вывести максимально возможное значение силы, которая может быть у стаи драконов из N голов.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 6 | 9 |
| 2 | 8 | 18 |
| 3 | 13 | 108 |

## 043. Нули

*(Время: 1 сек. Память: 16 Мб Сложность: 16%)*

Требуется найти самую длинную непрерывную цепочку нулей в последовательности нулей и единиц.

**Входные данные**

В единственной строке входного файла INPUT.TXT записана последовательность нулей и единиц (без пробелов). Суммарное количество цифр не превышает 100.

**Выходные данные**

В единственную строку выходного файла OUTPUT.TXT нужно вывести искомую длину цепочки нулей.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 00101110000110 | 4 |

**Разбор**

*Решение №1*

Здесь можно определить некоторую переменную-счетчик, в которой хранить текущее количество непрерывно идущих нулей при просмотре последовательности слева направо (можно и справа налево). При этом, если встречается единица, то нужно сбросить значение счетчика на ноль. Просматривая список нужно сверять значение счетчика с максимальным ранее найденным и таким образом запоминать максимальное. Считывать данные можно как в одну строку, обрабатывая нули и единицы как элементы массива, либо читать посимвольно до окончания потока данных.

```
c=max=0;
while(not EOF){
  read(x);
  if(x=='0') c++; else c=0;
  if(c>max) max=c;
}
write(max);
```

*Решение №2*

Можно использовать другой подход к решению этой задачи. Проверить наличие подпоследовательности из k нулей в исходной строке можно с помощью функции strstr в Си или pos в Паскале. Используя это можно создать некоторую пустую строку, которую последовательно удлинять на один ноль и каждый раз проверять имеется ли такая подстрока в исходной до тех пор, пока такая подстрока существует. Длина последней существующей подстроки нулей и будет ответом.

```
p='';
read(s);
while(p содержится в s) s=s+'0';
write(len(s)-1);
```

## 044. Стрелки

*(Время: 1 сек. Память: 16 Мб Сложность: 20%)*

Задана последовательность, состоящая только из символов ‘>’, ‘<’ и ‘-‘. Требуется найти количество стрел, которые спрятаны в этой последовательности. Стрелы – это подстроки вида ‘>>-->’ и ‘<--<<’.

**Входные данные**

В первой строке входного файла INPUT.TXT записана строка, состоящая из символов ‘>’, ‘<’ и ‘-‘ (без пробелов). Строка состоит не более, чем из 250 символов.

**Выходные данные**

В единственную строку выходного файла OUTPUT.TXT нужно вывести искомое количество стрелок.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | <<<<>>--><--<<--<<>>>--><<<<< | 4 |

## 045. Произведение цифр

*(Время: 1 сек. Память: 16 Мб Сложность: 47%)*

Требуется найти наименьшее натуральное число Q такое, что произведение его цифр равно заданному числу N.

**Входные данные**

В единственной строке входного файла INPUT.TXT записано одно целое число N ($0 \le N \le 10^9$).

**Выходные данные**

В выходной файл OUTPUT.TXT нужно вывести искомое число Q. В том случае, если такого числа не существует, следует вывести -1.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 10 | 25 |
| 2 | 13 | -1 |
| 3 | 90 | 259 |

## 046. Число E

*(Время: 1 сек. Память: 16 Мб Сложность: 10%)*

Выведите в выходной файл округленное до n знаков после десятичной точки число E. Число E, округленное до 25 знаков после десятичной точки, равно 2.7182818284590452353602875.

**Входные данные**

Входной файл INPUT.TXT содержит целое число n ($0 \le n \le 25$).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите ответ на задачу.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 0 | 3 |
| 2 | 25 | 2.718281828459045235360287 |
| 3 | 13 | 2.7182818284590 |

## 047. Наилучший делитель

*(Время: 1 сек. Память: 16 Мб Сложность: 26%)*

Будем говорить, что число a лучше числа b, если сумма цифр a больше суммы цифр числа b, а в случае равенства сумм их цифр, если число a меньше числа b. Например, число 124 лучше числа 123, так как у первого из них сумма цифр равна семи, а у второго – шести. Также, число 3 лучше числа 111, так как у них равны суммы цифр, но первое из них меньше.

Дано число n. Найдите такой его делитель (само число n и единица считаются делителями числа n), который лучше любого другого делителя числа n.

**Входные данные**

Первая строка входного файла содержит целое число n ($1 \le n \le 10^5$).

**Выходные данные**

В выходной файл выведите ответ на задачу.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 10 | 5 |
| 2 | 239 | 239 |

## 048. Наихудший делитель

*(Время: 1 сек. Память: 16 Мб Сложность: 23%)*

Будем говорить, что число a лучше числа b, если сумма цифр a больше суммы цифр числа b, а в случае равенства сумм их цифр, если число a меньше числа b. Например, число 124 лучше числа 123, так как у первого из них сумма цифр равна семи, а у второго – шести. Также, число 3 лучше числа 111, так как у них равны суммы цифр, но первое из них меньше.

Дано число n. Найдите такой его делитель d (само число n и единица считаются делителями числа n), что любой другой делитель c числа n лучше, чем d.

**Входные данные**

Первая строка входного файла INPUT.TXT содержит целое число n ($1 \le n \le 10^{5000}$).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите ответ на задачу.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 10 | 10 |
| 2 | 239 | 1 |

## 049. Шаблоны

*(Время: 1 сек. Память: 16 Мб Сложность: 35%)*

Шаблоном размера n назовем строку длины n, каждый из символов которой входит в множество {0,1,2,3,4, 5,6,7,8,9, a, b, c, d, e, f, g, ?}. Шаблоны преобразуются в строки из цифр по следующим правилам:

• символы от 0 до 9 могут быть преобразованы только сами в себя;
• символ a может преобразован в любой из символов 0,1, 2, 3;
• символ b может преобразован в любой из символов 1,2,3,4;

• символ c может преобразован в любой из символов 2,3,4,5;
• символ d может преобразован в любой из символов 3,4,5,6;
• символ e может преобразован в любой из символов 4,5,6,7;
• символ f может преобразован в любой из символов 5,6,7,8;
• символ g может преобразован в любой из символов 6,7,8,9;
• символ ? может преобразован в любой из символов от 0 до 9;

Даны два шаблона: p1 и p2. Рассмотрим множество S1 строк, которые могут быть получены из p1 по описанным правилам, и множество Sw строк, которые могут быть получены из p2. Необходимо найти количество строк, входящих в оба этих множества.

**Входные данные**

Первая строка входного файла INPUT.TXT содержит шаблон p1, вторая – шаблон p2. Шаблоны имеют одинаковый положительный размер, не больше 9.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите ответ на задачу.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | ???<br>abc | 64 |
| 2 | ???<br>000 | 1 |
| 3 | abc<br>999 | 0 |

## 050. Строки

*(Время: 1 сек. Память: 16 Мб Сложность: 44%)*

Циклическим сдвигом строки s называется строка $s_k s_{k+1} s_{k+2} \ldots s_{|s|} s_1 s_2 \ldots s_{k-1}$ для некоторого k, здесь |s| - длина строки s. Подстрокой строки s называется строка $s_i s_{i+1} \ldots s_{j-1} s_j$ для некоторых i и j. Вам даны две строки a и b. Выведите количество подстрок строки a, являющихся циклическими сдвигами строки b.

**Входные данные**

Первая строка входного файла INPUT.TXT содержит строку a ($1 \le |a| \le 1000$). Во второй строке входного файла записана строка b ($1 \le |b| \le \min(100,|a|)$). Обе строки состоят только из символов латинского алфавита и цифр.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите целое число – ответ на задачу.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | abcabc<br>abc | 4 |
| 2 | abcabc<br>acb | 0 |
| 3 | aaaaaaa<br>aa | 6 |
| 4 | aAaa8aaAa<br>aAa | 4 |

## 051. Факториалы!!!

*(Время: 1 сек. Память: 16 Мб Сложность: 26%)*

Определение 1:

n!!...!=n(n-k)(n-2k)...(n mod k), если n не делится на k,

n!!...!=n(n-k)(n-2k)...k, если n делится на k (знаков ! в обоих случаях k штук).

Определение 2:

X mod Y – остаток от деления X на Y.

Например, 10 mod 3 = 1; 3! = 3•2•1; 10!!! = 10•7•4•1;

Мы по заданным n и k смогли вычислить значение выражения из определения 1. А вам слабо?

**Входные данные**

Во входном файле INPUT.TXT содержится ровно одна строка. Сначала – целое число n ($1 \le n \le 10$), затем ровно один пробел, затем k восклицательных знаков ($1 \le k \le 20$).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите одно число – значение n!!..! .

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 10 !!! | 280 |
| 2 | 9 !! | 945 |
| 3 | 3 ! | 6 |

## 052. Счастливый билет

*(Время: 1 сек. Память: 16 Мб Сложность: 12%)*

Вы пользуетесь общественным транспортом? Вероятно, вы расплачивались за проезд и получали билет с номером. Счастливым билетом называют такой билет с шестизначным номером, где сумма первых трех цифр равна сумме последних трех. Т.е. билет с номером 385916 – счастливый, т.к. 3+8+5=9+1+6. Вам требуется написать программу, которая проверяет счастливость билета.

**Входные данные**

В единственной строке входного файла INPUT.TXT записано одно целое число N ($0 \le N < 10^6$).

**Выходные данные**

В выходной файл OUTPUT.TXT нужно вывести «YES», если билет с номером N счастливый и «NO» в противном случае.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 385916 | YES |
| 2 | 123456 | NO |

**Разбор**

Эта простая классическая задача имеет не единственное решение. Рассмотрим два решения.

*Решение №1*. Считаем счастливый билет в целочисленную переменную x и разобьем это число по цифрам так, чтобы первая цифра оказалась в переменной x1, вторая в x2 и т.д. до x6. После чего для того, чтобы убедиться в том, что билет счастливый достаточно сравнить значения x1+x2+x3 и x4+x5+x6. Но как так разбить число на цифры? Для этого можно воспользоваться целочисленным делением и остатком от деления. Например, последняя цифра целого числа x всегда равна x%10 (остатку от деления x на 10), а первая цифра шестизначного числа x всегда равна x/100000 (целочисленному делению x на 100000). Подумайте - почему? С помощью целочисленного деления мы можем от целого числа отбросить любое количество цифр (n цифр), разделив его на $10^n$ целочисленно, а с помощью остатка от деления на 10 мы получаем последнюю цифру. Например, в нашем случае x3=x/1000%10.

*Решение №2*. Можно счастливый билет считать в строковую переменную и полагая, что в нем ровно 6 цифр работать с ними как с массивом символов. Билет будет счастливым, если сумма ASCII-кодов первых трех символов равна сумме ASCII-кодов последних трех. На самом деле вовсе не обязательно осуществлять перевод в реальные цифры. На языке Си даже не придется для этого использовать специальную функцию (как ord в Паскале).

## 053. Раскраска таблицы умножения

*(Время: 1 сек. Память: 16 Мб Сложность: 32%)*

Таблицей умножения назовем таблицу размера n строк на m столбцов, в которой на пересечении i-ой строки и j-ого столбца стоит число i·j (строки и столбцы нумеруются с единицы).

В одной из математических школ было решено провести педагогический эксперимент. Для того, чтобы ученикам было проще запоминать таблицу умножения, некоторые числа в ней будут покрашены в красный, некоторые – в синий, а некоторые – в зеленый цвет (оставшиеся числа будут черными).

Процесс покраски чисел можно условно разбить на четыре этапа. На первом этапе все числа красятся в черный цвет. На втором – все четные числа красятся в красный цвет, на третьем – все числа, делящиеся на 3, красятся в зеленый цвет, на четвертом – все числа, делящиеся на 5, красятся в синий цвет.

Директор школы хочет знать, какое количество картриджей для принтеров необходимо закупить для печати таблиц. Поэтому ему необходима информация о том, сколько чисел какого цвета будет в одной раскрашенной таблице умножения n на m. Напишите программу, решающую задачу подсчета соответствующих количеств.

**Входные данные**

Входной файл INPUT.TXT содержит два натуральных числа n и m ($1 \le n, m \le 1000$).

**Выходные данные**

В первой строке выходного файла OUTPUT.TXT выведите количество чисел, покрашенных в красный цвет, во второй – в зеленый, в третьей – в синий, в четвертой – в черный. Следуйте формату, приведенному в примерах.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 10 10 | RED : 21<br>GREEN : 39<br>BLUE : 36<br>BLACK : 4 |
| 2 | 5 2 | RED : 5<br>GREEN : 2<br>BLUE : 2<br>BLACK : 1 |

## 054. Теория игр

*(Время: 1 сек. Память: 16 Мб Сложность: 28%)*

Одним из интересных объектов, изучаемых в теории игр, являются так называемые антагонистические игры двух лиц. Такие игры характеризуются множеством X стратегий первого игрока, множеством Y стратегий второго игрока и функцией выигрыша K(x, y) (x из X, y из Y). Если множества стратегий X и Y конечны, то такую игру принято называть матричной, так как функцию выигрыша K в этом случае удобно задавать матрицей.

Рассмотрим матричную игру, в которой X = {1, …, n}, Y = {1, …, m}. Матрицу выигрышей обозначим символом K. Нижним значением игры назовем число $\max_{i=1..n}\min_{j=1..m} K_{ij}$ . Верхним значением игры назовем число $\min_{j=1..m}\max_{i=1..n} K_{ij}$. Отметим также, что игры, у которых нижнее и верхнее значение совпадают, называются играми с седловой точкой.

Задана матрица выигрышей K для некоторой матричной игры. Найдите ее верхнее и нижнее значение.

**Входные данные**

Первая строка входного файла INPUT.TXT содержит целые числа n и m ($1 \le n, m \le 100$). Далее следуют n строк по m чисел в каждой. j-ое число i-ой строки равно $K_{ij}$ . Все $K_{ij}$ по модулю не превосходят 1000.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите нижнее и верхнее значение игры через пробел.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3 3 4 -1 -3<br>-2 1 3<br>0 2 -3 | -2 2 |
| 2 | 3 4<br>-1 0 2 1<br>-2 0 1 0<br>2 1 -1 -2 | -1 1 |

## 055. Фонарики

*(Время: 1 сек. Память: 16 Мб Сложность: 31%)*

«Одна голова хорошо, а две лучше. Одна лампочка хорошо, а две лучше!» – подумал Миша, и решил собрать фонарик с двумя лампочками. Теперь он хочет узнать, насколько фонарик с двумя лампочками лучше, чем фонарик с одной. Для этого Миша посветил фонариком на стену, и каждая из лампочек осветила на ней круг.

Эффективность фонарика Миша хочет оценить через площадь освещенной части стены. Миша догадался измерить координаты центров освещенных кругов и их радиусы (которые оказались одинаковыми). Причем, площадь, освещаемая фонариком с одной лампочкой известна, т.к. описана в документации, прилагаемой к фонарику. Но что делать дальше он не знает. Напишите программу, которая поможет Мише.

**Входные данные**

В первых двух строчках входного файла INPUT.TXT содержатся координаты $(x_1, y_1)$ и $(x_2, y_2)$ – центры кругов от лампочек собранного Мишей фонарика. В третьей строке задан радиус r описанных выше кругов, а четвертая строка содержит площадь освещения s фонариком из одной лампочки. Все числа целые и удовлетворяют следующим ограничениям: $1 \le x_1, y_1, x_2, y_2, r \le 100$, $1 \le s \le 10^5$. Так же заметим, что площади, освещаемые разными фонариками, отличаются друг от друга более чем на $10^{-3}$.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите «YES», если Мишин фонарик лучше старого (т.е. освещает большую площадь) и «NO» в противном случае.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 1 2<br>3 4<br>2<br>22 | YES |
| 2 | 1 1<br>100 100<br>1<br>7 | NO |

## 056. Jивой Jурнал

*(Время: 1 сек. Память: 16 Мб Сложность: 29%)*

Программист Саша участвует в создание блог-сервиса Jивой Jурнал. Планируется, что этот сервис будет предоставлять гораздо больше возможностей, чем известный всем LiveJournal. В настоящее жевремя проблему составляет реализация всех базовых возможностей LiveJournal'a. Одной из таких возможностей является поддержка списков друзей для пользователей.

Заданы: список пользователей, являющихся друзьями данного пользователя, и список пользователей, у которых данный пользователь содержится в списке друзей.

Необходимо получить список друзей данного пользователя (Friends), список его взаимных друзей (Mutual Friends), и список тех пользователей, у кого данный пользователь содержится в списке друзей, но которые не являются его взаимными друзьями (Also Friend of).

**Входные данные**

Первая строка входного файла INPUT.TXT содержит число n ($0 \le n \le 200$) друзей данного пользователя. Последующие n строк содержат каждая по одному имени пользователя, который является другом данного. (n + 2)-ая строка содержит число m ($0 \le m \le 200$) пользователей, у которых данный содержится в списке друзей. Далее заданы имена пользователей, у которых данный находится в списке друзей. Эти пользователи заданы в том же формате, что и друзья данного.

Имена пользователей – строки длиной не более 20 символов, содержащие только строчные буквы латинского алфавита и символы тире («-»). Каждый пользователь указан не более одного раза в каждом из списков.

**Выходные данные**

В выходной файл OUTPUT.TXT следует вывести список друзей данного пользователя (Friends), список его взаимных друзей (Mutual Friends), и список тех пользователей, у кого данный пользователь содержится в списке друзей, но которые не являются его взаимными друзьями (Also Friend of). В каждом списке пользователи должны быть отсортированы по алфавиту. Следуйте формату, приведенному в примерах.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3<br>vasya-pupkin<br>bill-hates<br>ivan-ivanov 2 vasya-pupkin<br>destroyer | Friends: bill-hates, ivan-ivanov, vasya-pupkin<br>Mutual Friends: vasya-pupkin<br>Also Friend of: destroyer |
| 2 | 0<br>0 | Friends:<br>Mutual Friends:<br>Also Friend of: |

## 057. Компьютерная сеть

*(Время: 1 сек. Память: 16 Мб Сложность: 33%)*

Компания «Маша и медведи» является самым крупным интернет-провайдером во всем лесу. Именно поэтому, с просьбой подключить их к интернету обратились N поросят. Домики поросят расположены в различных точках ($x_i$, $y_i$). Ближайшая точка подключения расположена в точке ($x_{net}$, $y_{net}$).

Для того чтобы подключиться к сети всем N поросятам необходимо:

1. провести провод от точки подключения до домика одного из поросят;
2. от подключенного поросенка провести провода ко всем остальным.

При этом провода могут при необходимости пересекаться.

Поросята платят деньги в зависимости от длины провода. Количество денег у них ограничено и составляет p тугриков. Они хотят определить: хватит ли им денег на подключение? Так же известно, что единица длины провода стоит c тугриков. Помогите им сделать необходимые расчеты!

**Входные данные**

В первой строке входного файла INPUT.TXT находится числа N, c и p – целые числа со следующими ограничениями: $1 \le N \le 10^3$, $1 \le c \le 10^4$, $1 \le p \le 10^9$ . В следующих N строках находятся координаты домов поросят ($x_i$; $y_i$). В последней строке записаны координаты точки соединения ($x_{net}$, $y_{net}$). Все координаты целые и не превосходят 1000 по модулю.

**Выходные данные**

В выходной файл OUTPUT.TXT следует вывести «YES», если у поросят достаточно денег для подключения и «NO» в противном случае.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3 2 6<br>0 0<br>1 0<br>0 1<br>-1 0 | YES |
| 2 | 3 1 5<br>1 1<br>2 2<br>3 3<br>4 4 | NO |

## 058. Проверка на симпатичность

*(Время: 1 сек. Память: 16 Мб Сложность: 28%)*

Рассмотрим таблицу, содержащую n строк и m столбцов, в каждой клетке которой расположен ноль или единица. Назовем такую таблицу симпатичной, если в ней нет ни одного квадрата 2 на 2, заполненного целиком нулями или целиком единицами.

Так, например, таблица 4 на 4, расположенная слева, является симпатичной, а расположенная справа таблица 3 на 3 – не является.

| | | | |
|---|---|---|---|
| 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 0 | 0 | 0 |

| | | |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 0 | 1 |
| 1 | 1 | 1 |

Задано несколько таблиц. Необходимо для каждой из них выяснить, является ли она симпатичной.

**Входные данные**

Первая строка входного файла INPUT.TXT содержит количество t ($1 \le t \le 10$) наборов входных данных. Далее следуют описания этих наборов. Описание каждого набора состоит из строки, содержащей числа n и m ($1 \le n, m \le 100$), и n строк, каждая из которых содержит по m чисел, разделенных пробелами. j-ое число в i+1-ой строке описания набора входных данных – элемент $a_{ij}$ соответствующей таблицы. Гарантируется, что все $a_{ij}$ равны либо нулю, либо единице.

**Выходные данные**

Для каждого набора входных данных выведите в файл OUTPUT.TXT единственную строку, содержащую слово «YES», если соответствующая таблица является симпатичной, и слово «NO» – в противном случае.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3<br>1 1<br>0<br>4 4<br>1 0 1 0<br>1 1 1 0<br>0 1 0 1<br>0 0 0 0<br>3 3<br>0 0 1<br>0 0 1<br>1 1 1 | YES<br>YES<br>NO |

**Разбор**

В этой задаче необходимо последовательно считывать в двумерный массив все представленные матрицы и проверять их на симпатичность, результат проверки выводить в выходной файл. Для проверки текущей матрицы на симпатичность можно в двойном цикле перебрать всевозможные подмассивы 2x2 и проверить: существует ли среди них хотя бы один, состоящий из одинаковых элеменов. Если да, то в файл нужно вывести «NO» и «YES» в противном случае. Механизм проверки одной матрицы на симпатичность можно описать следующим образом:

```
Ok=true;
for i=1..n-1{
  for j=1..m-1{
    if( (a[i][j]+a[i][j+1]+a[i+1][j]+a[i+1][j+1]) mod 4 == 0 ) Ok=false;
  }
}
if(Ok) write("YES") else write("NO");
```

Следует заметить, что использование двумерного массива вовсе не обязательно. Здесь не обязательно запоминать все элементы матрицы, достаточно помнить предыдущую и текущую строчку и в процессе считывания данных проверять подмассивы 2x2. Такой алгоритм немного сложнее для реализации, но более экономичен по используемой памяти, что иногда не менее важно.

## 059. Несложное вычисление

*(Время: 1 сек. Память: 16 Мб Сложность: 25%)*

Задано натуральное число n. Необходимо перевести его в k-ичную систему счисления и найти разность между произведением и суммой его цифр в этой системе счисления.

Например, пусть n = 239, k = 8. Тогда представление числа n в восьмеричной системе счисления – 357, а ответ на задачу равен $3 \times 5 \times 7 - (3 + 5 + 7) = 90$.

**Входные данные**

Входной файл INPUT.TXT содержит два натуральных числа: n и k ($1 \le n \le 10^9$, $2 \le k \le 10$). Оба этих числа заданы в десятичной системе счисления.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите ответ на задачу (в десятичной системе счисления).

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 239 8 | 90 |
| 2 | 1000000000 7 | -34 |

**Разбор**

Алгоритм решения задачи похож на перевод числа n в k-ую систему счисления. При этом в процессе вычисления каждую полученную цифру можно прибавлять к некоторой переменной sum, а так же домножать к переменной mult. В результате в sum получим сумму, а в mult произведение. Останется только вывести их разность.

Представим вышеопианный алгоритм в следующей форме:

```
read(n,k);
sum=0; mult=1;
while(n>0){
    sum = sum + n mod k;
    mult = mult * (n mod k);
    n = n div k;
}
write(mult-sum);
```

## 060. Сверхпростые числа

*(Время: 1 сек. Память: 16 Мб Сложность: 39%)*

Простым числом будем называть натуральное число, большее единицы и делящееся только на единицу и на само себя. Выпишем все простые числа в порядке возрастания и i-ое в этом порядке число обозначим pi (число 2 при этом будет иметь номер 1). Так, например, $p_1 = 2$, $p_2 = 3$, $p_3 = 5$, $p_{52} = 239$.

Скажем, что число $p_i$ является сверхпростым, если $i = p_k$ для некоторого k. Иными словами, сверхпростое число – это простое число, номер которого в списке простых чисел, упорядоченном по возрастанию, является простым числом.

Дано натуральное число k. Упорядочим все сверхпростые числа по возрастанию. Найдите k-ое сверхпростое число в этом порядке.

**Входные данные**

Входной файл INPUT.TXT содержит натуральное число k ($1 \le k \le 500$).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите k-ое сверхпростое число.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 1 | 3 |
| 2 | 2 | 5 |
| 3 | 3 | 11 |
| 4 | 100 | 3911 |

## 061. Баскетбол

*(Время: 1 сек. Память: 16 Мб Сложность: 5%)*

Известны результаты каждой из 4х четвертей баскетбольной встречи. Нужно определить победителя матча.

**Входные данные**

Входной файл INPUT.TXT содержит 4 строки, в каждой строке находится два целых числа a и b – итоговый счет в соответствующей четверти. a – количество набранных очков за четверть первой командой, b – количество очков набранных за четверть второй командой ($0 \le a, b \le 100$).

**Выходные данные**

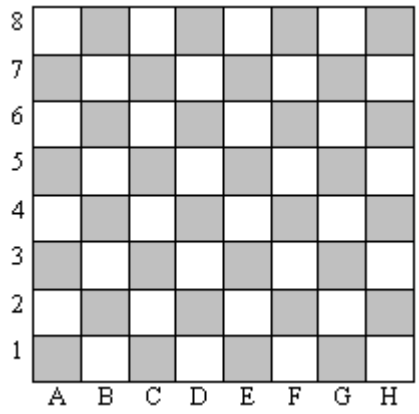
В выходной файл OUTPUT.TXT выведите номер выигравшей команды, в случае ничьей следует вывести «DRAW».

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 26 17<br>13 15<br>19 11<br>14 16 | 1 |
| 2 | 14 15<br>17 18<br>20 20<br>15 17 | 2 |
| 3 | 15 16<br>18 17<br>10 12<br>14 12 | DRAW |

## 062. Клетки

*(Время: 1 сек. Память: 16 Мб Сложность: 15%)*




Известно, что шахматная доска имеет размерность 8x8 и состоит из клеток 2х цветов, например, черного и белого (см. рисунок). Каждая клетка имеет координату, состоящую из буквы и цифры. Горизонтальное расположение клетки определяется буквой от A до H, а вертикальное – цифрой от 1 до 8. Заметим, что клетка с координатой A1 имеет черный цвет. Требуется по заданной координате определить цвет клетки.

**Входные данные**

В единственной строке входного файла INPUT.TXT записана координата клетки на шахматной доске: всего два символа – буква и цифра (без пробелов).

**Выходные данные**

В выходной файл OUTPUT.TXT нужно вывести «WHITE», если указанная клетка имеет белый цвет и «BLACK», если она черная.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | C3 | BLACK |
| 2 | G8 | WHITE |

## 063. Загадка

*(Время: 1 сек. Память: 16 Мб Сложность: 18%)*

Петя и Катя – брат и сестра. Петя – студент, а Катя – школьница. Петя помогает Кате по математике. Он задумывает два натуральных числа X и Y (X, Y≤1000), а Катя должна их отгадать. Для этого Петя делает две подсказки. Он называет сумму этих чисел S и их произведение P. Помогите Кате отгадать задуманные Петей числа.

**Входные данные**

Входной файл INPUT.TXT содержит два натуральных числа S и P, разделенных пробелом.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите два числа X и Y, загаданные Петей. Числа следует вывести в порядке неубывания своих значений, разделенные пробелом.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 4 4 | 2 2 |
| 2 | 5 6 | 2 3 |

**Разбор**

Эту задачу можно решить, составив систему из двух уравнений и двух неизвестных:

$$X+Y=S$$
$$X*Y=P$$

Выразив X из первого уравнения и подставив во второе мы получим квадратное уравнение, из которого получим X, а значит и Y. Но это решение не самое быстрое для реализации. Здесь можно использовать то, что компьютер способен выполнять миллионы простых операций в секунду. Поэтому можно реализовать полный перебор всех вариантов значений X и Y (очевидно, всевозможных значений получится не более миллиона). Причем, для удобства можно сразу положить что X≤Y. Таким образом, мы сократим перебор вариантов вдвое и сможем найти однозначное решение, которое без проблем возможно вывести в порядке неубывания. Следующий алгоритм реализует вышеописанные рассуждения:

```
read(s,p);
for x=1..1000{
  for y=x..1000{
    if(x+y==s и x*y==p) write(x,' ',y);
  }
}
```

Полезно уметь оценивать временную сложность задачи. Несмотря на то, что ЭВМ способна на многое, далеко не любая задача в случае неэффективного ее решения сможет уложиться в требуемые временные ограничения. Например, если бы дипазон чисел X и Y был в диапазоне до 30000, то такое решение с полным перебором привело бы к провалу, т.к. в этом случае возможно потребовалось бы выполнение порядка миллиарда операций, что вполне сомнительно для ограничений в 1 секунду. Но и в этом случае перебор был бы возможен. Действительно, когда значение X определено, то не обязательно перебирать всевозможные значения Y, т.к. Y=S-X. Поэтому более короткий и быстрый алгоритм решения этой задачи может выглядеть следующим образом:

```
read(s,p);
for x=1..30000{
  y=s-x;
  if(x<=y и x*y==p) write(x,' ',y);
}
```

## 064. Простой ряд

*(Время: 1 сек. Память: 16 Мб Сложность: 27%)*

Простым числом называется натуральное число (большее 1), которое делится нацело только на 1 и на само себя. Например, числа 2, 3, 5, 7, 11 и 23 – простые. Назовем простым рядом последовательность цифр, полученную следующим образом: подряд идущие по возрастанию простые числа (начиная с 2) записываются друг за другом. Начало данного ряда выглядит так: 23571113171923… Необходимо найти цифру, стоящую в простом ряду на указанном месте. Нумерация позиций начинается с единицы.

**Входные данные**

В первой строке входного файла INPUT.TXT записано натуральное число M – количество тестов. Во второй строке записано M чисел через пробел, каждое число – номер позиции в простом ряду ($1 \le M \le 1000$, $1 \le$ номер позиции $\le 10000$).

**Выходные данные**

В выходной файл OUTPUT.TXT для каждой позиции выведите цифру из простого ряда, стоящую на этой позиции. Вывести следует M цифр в одной строке, не разделяя цифры пробелами.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3<br>1 4 11 | 271 |
| 2 | 5<br>2 5 6 8 12 | 31139 |

## 065. Расстояние Хэмминга

*(Время: 1 сек. Память: 16 Мб Сложность: 32%)*

В связи с особенностями линии связи, используемой для передачи сообщений из пункта A в пункт B, каждый бит принятого сообщения с вероятностью 0.001 содержит ошибку.

Из пункта A в пункт B было послано одно из n сообщений $m_1$, $m_2$, ..., $m_n$. В пункте B было принято сообщение s.

Ваша задача заключается в определении наиболее вероятного исходного сообщения. Очевидно, что оно будет одним из тех сообщений, расстояние Хэмминга между которым и строкой s минимально.

Расстоянием Хэмминга двух строк a и b одинаковой длины называется количество позиций, в которых эти строки различаются (количество элементов в множестве $\{i \mid 1 \le i \le |a|, a_i \ne b_i\}$).

**Входные данные**

Первая строка входного файла INPUT.TXT содержит s – принятое сообщение. Вторая строка содержит целое число n – количество сообщений, которые могли быть отправлены. Следующие n строк содержат $m_i$ – эти сообщения. Длины всех сообщений равны ($|s| = |m_1| = |m_2| = ... = |m_n|$). Сообщения состоят только из символов 0 и 1. Размер входного файла не превосходит 60 Кб.

**Выходные данные**

В первую строку выходного файла OUTPUT.TXT выведите k – количество сообщений, на которых достигается минимум расстояния Хэмминга. Во вторую строку выведите в порядке возрастания k чисел – номера этих сообщений.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 010101<br>3<br>110011<br>011001<br>000111 | 2<br>2 3 |

## 066. Клавиатура

*(Время: 1 сек. Память: 16 Мб Сложность: 11%)*

Для данной буквы латинского алфавита нужно вывести справа стоящую букву на стандартной клавиатуре. При этом клавиатура замкнута, т.е. справа от буквы «p» стоит буква «a», от буквы «l» стоит буква «z», а от буквы «m» – буква «q».

**Входные данные**

Входной файл INPUT.TXT содержит один символ – маленькую букву латинского алфавита.

**Выходные данные**

В выходной файл OUTPUT.TXT следует вывести букву стоящую справа от заданной буквы, с учетом замкнутости клавиатуры.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | q | w |
| 2 | t | y |
| 3 | p | a |
| 4 | l | z |
| 5 | m | q |

## 067. Маска подсетей

*(Время: 1 сек. Память: 16 Мб Сложность: 33%)*

Рассмотрим компьютерную сеть с настроенной TCP/IP маршрутизацией. Будем рассматривать некоторую ее модификацию. А именно в этой сети находить N подсетей. Каждая подсеть характеризуется своей маской. Маска подсети представляет собой 4 однобайтных числа, разделенных точкой. Причем для масок выполнено следующее свойство: если представить маску в двоичном виде, то сначала она будет содержать k единиц, а потом q нулей, причем k + q = 32. Например, 255.255.255.0 – маска подсети, а 192.168.0.1 – нет.

Поясним, как получается двоичное представление IP-адреса. Для этого числа, составляющие IP-адрес, представляются в двоичной системе счисления (при этом каждое из них дополняется ведущими нулями до длины в 8 цифр), после чего удаляются точки. Получившееся 32-битное число и есть двоичное представление IP-адреса. Например, для адреса 192.168.0.1 этот процесс выглядит так: 192.168.0.1 → 11000000.10101000.00000000.00000001 → 11000000101010000000000000000001. Таким образом, двоичным представлением IP-адреса 192.168.0.1 является 11000000101010000000000000000001.

Будем говорить, что два компьютера с $IP_1$ и $IP_2$ лежат в подсети, если $IP_1 \wedge Mask = IP_2 \wedge Mask$, где Mask – маска этой подсети, а $\wedge$ – операция побитового логического «и». IP компьютера представляет собой так же 4 однобайтных числа, разделенных точкой.

Вам даны M пар IP адресов компьютеров. Для каждой из них Вам надо определить, в скольких подсетях из заданных они лежат.

**Входные данные**

В первой строке входного файла INPUT.TXT записано число N – количество подсетей. В следующих N строках перечислены маски этих подсетей. В N + 2 строке находится число M ($0 \le M \le 10000$). В следующих M строках записаны пары IP адресов, разделенных пробелом.

**Выходные данные**

Для каждой пары IP адресов в отдельной строке выходного файла OUTPUT.TXT выведите количество подсетей, в которых лежат оба компьютера.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 2<br>255.255.255.255<br>255.255.255.0<br>3<br>192.168.31.1 192.168.31.2<br>192.168.31.3 192.168.31.4<br>192.168.31.1 192.167.31.2 | 1<br>1<br>0 |

## 068. Дом - Школа - Дом

*(Время: 1 сек. Память: 16 Мб Сложность: 21%)*

Мальчик Вася каждый день ездит на метро. Утром он едет в школу, а вечером того же дня, обратно из школы, домой. Для того чтобы немного сэкономить, он покупает электронную смарт-карту на X поездок. Когда он хочет зайти в метро, он прикладывает карту к турникету. Если на карте осталось ненулевое количество поездок, то турникет пропускает Васю и списывает с карты одну поездку. Если же на карте не осталось поездок, то турникет не пропускает Васю, и он (Вася) вынужден купить на этой же станции новую карту на X поездок и вновь пройти через турникет.

Вася заметил, что в связи с тем, что утром метро переполнено, покупать новую карту утром накладно по времени, и он может опоздать в школу. В связи с этим он хочет понять: будет ли такой день, что с утра, поехав в школу, окажется, что у него на карточке ноль поездок.

Вася больше никуда на метро не ездит и поэтому заходит в метро только на станции около дома и на станции около школы.

**Входные данные**

Во входном файле INPUT.TXT содержится ровно 2 строки. В первой содержится слово «School» или «Home» в зависимости от того, где первый раз Вася купил карточку на X поездок. Во второй строке содержится натуральное число X, $1 \le X \le 1000$.

**Выходные данные**

В выходной файл OUTPUT.TXT следует вывести «Yes», если будет такой день, что утром у Васи на карточке окажется ноль поездок и «No» в противном случае.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | Home<br>1 | Yes |
| 2 | School<br>2 | No |

## 069. N-угольное колесо

*(Время: 1 сек. Память: 16 Мб Сложность: 28%)*

На одном известном автозаводе страны N-мерики главный инженер-рационализатор внес предложение вместо круглых колес использовать колеса в форме правильных N-угольников. «При этом, – сказал он, – важным показателем качества такого колеса будет разность между радиусом описанной окружности и радиусом вписанной окружности». Причем колесо считается качественным, если его показатель качества меньше единицы.

Задано число N и длина A стороны N-угольного колеса. Необходимо определить: является ли такое колесо качественным.

**Входные данные**

Входной файл INPUT.TXT содержит два натуральных числа: N и A ($3 \le N \le 1000$, $1 \le A \le 1000$).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите «YES», если это качественное колесо и «NO» в противном случае.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3 1 | YES |
| 2 | 239 566 | NO |

## 070. Степень строки

*(Время: 1 сек. Память: 16 Мб Сложность: 30%)*

Пусть задана строка $s = s_1s_2...s_n$. Назовем ее k-ой ($k > 0$) степенью $s^k$ строку $s^k = s_1s_2 \ . \ . \ .s_ns_1s_2 \ . \ . \ .s_n.....s_1s_2...s_n$ (k раз). Например, третьей степенью строки abc является строка abcabcabc.

Корнем k степени из строки s называется такая строка t (если она существует), что $t^k = s$.

Ваша задача состоит в том, чтобы написать программу, находящую степень строки или корень из нее.

**Входные данные**

Первая строка входного файла INPUT.TXT содержит строку s, она содержит только маленькие буквы латинского алфавита и имеет ненулевую длину, не превосходящую 1000.

Вторая строка входного файла содержит целое число $k \ne 0$, $|k| < 100001$. Если $k > 0$, то необходимо найти k-ую степень строки s, если $k < 0$, то необходимо найти корень степени $|k|$ из s.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите строку, являющуюся ответом на задачу. Если длина ответа превосходит 1023 символа, выведите только первые 1023 символа. Если искомой строки не существует – выведите NO SOLUTION.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | Abc<br>3 | abcabcabc |
| 2 | Abcdabcd<br>-2 | abcd |
| 3 | Abcd<br>-4 | NO SOLUTION |

## 071. Две кучки камней

*(Время: 1 сек. Память: 16 Мб Сложность: 43%)*

У Вас есть N кучек камней с массами $W_1$, $W_2$ , … $W_N$. Требуется разложить камни на 2 кучки так, чтобы разница масс этих кучек была минимальной.

**Входные данные**

В первой строке входного файла INPUT.TXT записано число N – количество камней ($1 \le N \le 18$). Во второй строке через пробел перечислены массы камней $W_1$, $W_2$ , … $W_N$ ($1 \le W_i \le 10^5$).

**Выходные данные**

В единственную строку выходного файла OUTPUT.TXT нужно вывести одно неотрицательное целое число – минимально возможную разницу между массами двух кучек.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 5<br>5 8 13 27 14 | 3 |

## 072. Анаграмма

*(Время: 1 сек. Память: 16 Мб Сложность: 45%)*

Пусть задано некоторое слово, состоящее из букв английского алфавита длинной не более 80 символов (например, «WORD»). Рассмотрим набор возможных перестановок, состоящих из букв данного слова (например, «RDOW», «WODR» и т.д.). Требуется выбрать из этого множества слово, следующее по алфавиту за исходным.

**Входные данные**

В единственной строке входного файла INPUT.TXT записано слово, не последнее по алфавиту среди возможных его перестановок.

**Выходные данные**

В единственную строку выходного файла OUTPUT.TXT нужно вывести следующее слово по алфавиту.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | abdc | acbd |
| 2 | word | wrdo |

## 073. Расшифровка

*(Время: 1 сек. Память: 16 Мб Сложность: 28%)*

Рассмотрим работу простейшего шифра. Шифруемое сообщение состоит из латинских букв, записанных в нижнем регистре и символа пробела. Шифрование происходит посимвольно. Каждой букве ставим в соответствие число: a – 1, b – 2, … , z – 26, ‘ ‘ – 27. Далее индекс символа складывается с номером в сообщении по модулю 27, а результат сложения представляется в системе счисления с основанием 27 (0, 1, …, Q в верхнем регистре).

Необходимо написать дешифратор.

**Входные данные**

В единственной строке входного файла INPUT.TXT содержится закодированная строка, длиной не более 255 символов. Строка записана в верхнем регистре.

**Выходные данные**

В единственную строку выходного файла OUTPUT.TXT нужно вывести расшифровку заданной строки в виде символов латинского алфавита в нижнем регистре.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | L7MO | test |
| 2 | 576J9FLF | decoding |

## 074. Прыжки с шестом

*(Время: 1 сек. Память: 16 Мб Сложность: 38%)*

В соревнованиях по прыжкам с шестом было замечено одно интересное явление: на очередном этапе соревнований успешные и неуспешные попытки прыжков чередовались: успешный, неуспешный, успешный, неуспешный и т.д. (первый был успешным). Спортсменам разрешалась только одна попытка. Тот, кто преодолевал планку, переходил в следующий тур (этап), а тот, кто делал неудачную попытку – выбывал из соревнований. Таким образом, первым выбывал всегда спортсмен с номером 2, а последним – победитель с номером 1.

Требуется написать программу, которая по количеству участников и номеру спортсмена вычислит, каким по счету данный спортсмен выбыл из соревнований.

**Входные данные**

В единственной строке входного файла INPUT.TXT содержатся два натуральных числа: общее число спортсменов N и порядковый номер спортсмена в стартовом списке M. Числа разделены пробелом ($1 \le M, N \le 10^9$).

**Выходные данные**

В единственную строку выходного файла OUTPUT.TXT нужно вывести каким по счету спортсмен M выбыл из соревнований. Если это победитель состязания, то выводится число N.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 4 2 | 1 |
| 2 | 4 1 | 4 |
| 3 | 9 5 | 7 |

## 075. Сумма произведений

*(Время: 1 сек. Память: 16 Мб Сложность: 65%)*

Требуется вычислить сумму произведений всех N-значных чисел. При этом следует учесть, что если в числе встречается цифра 0, то произведение его цифр равно нулю. Для N=3 искомая сумма представлена следующим рядом:

S = 1*0*0 + 1*0*1 + 1*0*2 + … + 9*9*8 + 9*9*9 = 91125.

**Входные данные**

В единственной строке входного файла INPUT.TXT записано натуральное число N (N < 1000).

**Выходные данные**

В единственную строку выходного файла OUTPUT.TXT нужно вывести одно целое число – сумму произведений всех N-значных чисел.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 1 | 45 |
| 2 | 3 | 91125 |
| 3 | 5 | 184528125 |

## 076. Музей

*(Время: 1 сек. Память: 16 Мб Сложность: 59%)*

В музее регистрируется в течение суток время прихода и ухода каждого посетителя. Таким образом, за день получены N пар значений, где первое значение в паре показывает время прихода посетителя и второе значение - время его ухода. Требуется найти максимальное число посетителей, которые находились в музее одновременно.

**Входные данные**

В первой строке входного файла INPUT.TXT записано натуральное число N ($N < 10^5$) – количество зафиксированных посетителей в музее в течении суток. Далее, идут N строк с информацией о времени визитов посетителей: в каждой строке располагается отрезок времени посещения в формате «ЧЧ:ММ ЧЧ:ММ» (00:00 ≤ ЧЧ:ММ ≤ 23:59).

**Выходные данные**

В единственную строку выходного файла OUTPUT.TXT нужно вывести одно целое число – максимальное количество посетителей, одновременно находящихся в музее.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 6<br>09:00 10:07<br>10:20 11:35<br>12:00 17:00<br>11:00 11:30<br>11:20 12:30<br>11:30 18:15 | 4 |

## 077. Нолики

*(Время: 1 сек. Память: 16 Мб Сложность: 63%)*

Для заданных натуральных чисел N и K требуется вычислить количество чисел от 1 до N, имеющих в двоичной записи ровно K нулей.

Например, если N=8 и K=1, то мы можем записать все числа от 1 до 8 в двоичной системе счисления: 1, 10, 11, 100, 101, 110, 111 и 1000.

Откуда видно, что только числа 10, 101 и 110 имеют ровно один ноль в записи, т.е. правильный ответ – 3.

**Входные данные**

В единственной строке входного файла INPUT.TXT записано два натуральных числа через пробел N и K, не превышающих $10^9$.

**Выходные данные**

В единственную строку выходного файла OUTPUT.TXT нужно вывести одно целое число – количество чисел от 1 до N с K нулями в двоичном представлении.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 8 1 | 3 |
| 2 | 13 2 | 4 |
| 3 | 1000 5 | 210 |

## 078. Бутылки

*(Время: 1 сек. Память: 16 Мб Сложность: 48%)*

Группа программистов собралась в понедельник и на все свои деньги купила «Sprite» в бутылках емкостью по 0.25 л., не забыв взять сдачу.

Во вторник они сдали пустую посуду, добавили оставшуюся сдачу и вновь купили столько таких же бутылок «Sprite», сколько могли.

Так они действовали до пятницы. В пятницу, сдав посуду и добавив сдачу с четверга, они смогли купить только одну бутылку напитка. При этом денег у них уже не осталось.

Требуется написать программу, определяющую минимальную сумму, которой располагали программисты в понедельник.

**Входные данные**

Входной файл INPUT.TXT состоит из единственной строки, содержащей два целых числа – F (стоимость одной бутылки «Sprite») и P (стоимость одной пустой бутылки из под «Sprite»), разделенных пробелом.

Ограничения: $1 \le P < F \le 10^9$, начальная сумма не превосходит $2*10^9$.

**Выходные данные**

В единственную строку выходного файла OUTPUT.TXT нужно вывести одно целое число – минимальную сумму, которой располагали программисты в понедельник.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 7 3 | 83 |

## 079. Последняя цифра A^B

*(Время: 1 сек. Память: 16 Мб Сложность: 21%)*

Требуется написать программу, которая находит цифру, на которую оканчивается число A^B.

**Входные данные**

Входной файл INPUT.TXT состоит из единственной строки, содержащей два целых числа A и B, разделенных пробелом ($1 \le A, B \le 10000$).

**Выходные данные**

В единственную строку выходного файла OUTPUT.TXT нужно вывести цифру, на которую оканчивается A^B.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 2 2 | 4 |
| 2 | 3 7 | 7 |
| 3 | 24 9 | 4 |

## 080. Тождество

*(Время: 1 сек. Память: 16 Мб Сложность: 26%)*

Вам необходимо проверить домашнюю работу Васи Пупкина, в которой он написал равенство. Например, запись вида «2+3=5» является правильной, а «23*7=421» неверная, но корректная. Корректной записью выражения будем называть последовательность: число, операция («+», «-», «*», «/»), число, знак равенства, число. Т.е. если в записи не хватает цифр или же встречается неизвестный символ. Например, записи «2*=3», «173» и «2+2=a» некорректны.

**Входные данные**

Входной файл INPUT.TXT состоит из единственной строки, содержащей запись арифметического выражения. Все числа в записи не превышают по абсолютной величине 30000. Длина арифметического выражения не превышает 100 символов.

**Выходные данные**

В выходной файл OUTPUT.TXT нужно вывести «YES», если указанная запись правильна (т.е. равенство представляет собой тождество), «NO» – если корректная, но неверная и «ERROR», если в записи присутствуют ошибки.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 2+3=5 | YES |
| 2 | 3*7=20 | NO |
| 3 | two plus three is five | ERROR |

## 081. Арбузы

*(Время: 1 сек. Память: 16 Мб Сложность: 14%)*

Иван Васильевич пришел на рынок и решил купить два арбуза: один для себя, а другой для тещи. Понятно, что для себя нужно выбрать арбуз потяжелей, а для тещи полегче. Но вот незадача: арбузов слишком много и он не знает как же выбрать самый легкий и самый тяжелый арбуз? Помогите ему!

**Входные данные**

В первой строке входного файла INPUT.TXT задано одно число N – количество арбузов. Вторая строка содержит N чисел, записанных через пробел. Здесь каждое число – это масса соответствующего арбуза. Все числа натуральные и не превышают 30000.

**Выходные данные**

В выходной файл OUTPUT.TXT нужно вывести два числа через пробел: массу арбуза, который Иван Васильевич купит теще и массу арбуза, который он купит себе.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 5<br>5 1 6 5 9 | 1 9 |

**Разбор**

В этой задаче из представленных n чисел следует выбрать наибольшее и наименьшее число, а затем просто их вывести. Для поиска максимального элемента следует определить переменную max, в которую предварительно можно занести наименьшее возможное значение, 0 например (ведь все числа натуральные, т.е. больше нуля). Далее следует считывать в переменную x текущее значение и проверять: не больше ли оно ранее найденного, которое как раз храниться в max. Если да, то будем записывать его в max. Таким образом, по окончании процесса мы получим в max наибольшее значение. Поиск минимального элемента происходит аналогично. Заметим, что в этой задаче можно обойтись без использования массивов.

Рассмотрим алгоритм решения этой задачи:

```
read(n);
max=0; min=32000;
for i=1..n{
  read(x);
  if(x>max) max=x;
  if(x<min) min=x;
}
write(min,' ',max);
```

На самом деле в этой задаче может возникнуть непонятный момент: что делать, когда арбуз всего один? Кому его нужно купить: себе или теще? Если рассуждать разумно (по жизни), то конечно лучше купить его себе, а теща обойдется и без арбуза. Но если следовать условиям задачи получается, что этот арбуз будет адресован как Ивану Васильевичу, так и его теще и в данном случае нужно выводить два одинаковых числа в качестве ответа. Впрочем, подобных тестов нет.

## 082. Пересечение множеств

*(Время: 1 сек. Память: 16 Мб Сложность: 34%)*

Даны два неупорядоченных набора натуральных чисел (может быть, с повторениями). Выдать без повторений в порядке возрастания все те числа, которые встречаются в обоих наборах.

**Входные данные**

В первой строке входного файла INPUT.TXT записано через пробел два целых числа N и M ($1 \le N, M \le 10^6$) – количество элементов первого и второго наборов, соответственно. В следующих строках записано сначала N чисел первого набора, а затем M чисел второго набора. Числа разделены пробелами или символами конца строки. Каждое из этих чисел попадает в промежуток от 0 до $10^5$.

**Выходные данные**

В выходной файл OUTPUT.TXT нужно записать в возрастающем порядке без повторений все числа, которые входят как в первый, так и во второй набор. Числа разделять одним пробелом. Если таких чисел нет, то выходной файл должен оставаться пустым.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 11 6<br>2 4 6 8 10 12 10 8 6 4 2<br>3 6 9 12 15 18 | 6 12 |

**Разбор**

Эта задача сложна своими ограничениями. Самое ресурсоемкое, но простое решение, которое может прийти в голову – это считать данные в массивы, исключить из них повторяющиеся элементы, далее пробегая по одному из них смотреть: есть ли он во втором и если есть, то выводить в выходной файл. Здесь только исключение повторяющихся элементов может при неэффективном решении иметь сложность $O(n^2)$, что для миллиона элементов может вызвать TLE (Time limit exceeded).

Рассмотрим теперь верное решение этой задачи. Здесь вовсе не нужны два массива, длина которых может быть до миллиона. В силу конечности диапазона возможных значений мы можем описать некоторый массив a[i] для $i=0..10^5$, в i-м значении будем хранить 0, если число i не содержится ни в одном из двух наборов чисел. Если число i имеется в первом наборе, то запишем туда 1, а если и в обоих наборах, то запишем туда 2. После чего пробегая по массиву нужно будет вывести все i, у которых a[i]=2. Заполнение массива a[i] происходит следующим образом: сначала все элементы обнуляются, далее при чтении элемента x первого набора можно выполнять присваивание a[x]=1. При просмотре элементов второго набора для каждого элемента y можно a[y]=2 только в том случае, когда a[y]=1.

Приведенный выше алгоритм можно формализовать к следующему виду:

```
int a[0..10^5]={0...0};

read(n,m);
for i=1..n{
  read(x);
  a[x]=1;
}
for i=1..m{
  read(y);
  if(a[y]==1) a[y]=2;
}
for i=1..10^5{
  if(a[i]==2) write(i,' ');
}
```

## 083. Симпатичные узоры

*(Время: 1 сек. Память: 16 Мб Сложность: 66%)*

Компания BrokenTiles планирует заняться выкладыванием во дворах у состоятельных клиентов узор из черных и белых плиток, каждая из которых имеет размер 1x1 метр. Известно, что дворы всех состоятельных людей имеют наиболее модную на сегодня форму прямоугольника MxN метров.

Однако при составлении финансового плана у директора этой организации появилось целых две серьезных проблемы: во-первых, каждый новый клиент, очевидно, захочет, чтобы узор, выложенный у него во дворе, отличался от узоров всех остальных клиентов этой фирмы, а во-вторых, этот узор должен быть симпатичным. Как показало исследование, узор является симпатичным, если в нем нигде не встречается квадрата 2x2 метра, полностью покрытого плитками одного цвета. На рисунке 1 показаны примеры различных симпатичных узоров, а на рисунке 2 – несимпатичных.

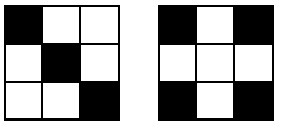


Рисунок 1

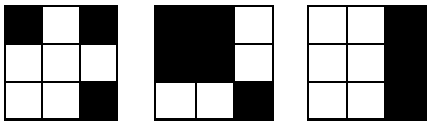


Рисунок 2

Для составления финансового плана директору необходимо узнать, сколько клиентов он сможет обслужить, прежде чем симпатичные узоры данного размера закончатся. Помогите ему!

**Входные данные**

В перовой строке входного файла INPUT.TXT находятся два положительных целых числа, разделенные пробелом – M и N ($1 \le M \cdot N \le 30$).

**Выходные данные**

Выведите в выходной файл OUTPUT.TXT единственное число – количество различных симпатичных узоров, которые можно выложить во дворе размера MxN. Узоры, получающиеся друг из друга сдвигом, поворотом или отражением считаются различными.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 2 2 | 14 |
| 2 | 3 3 | 322 |

## 084. Выпуклая оболочка

*(Время: 1 сек. Память: 16 Мб Сложность: 27%)*

Рассмотрим бесконечный лист клетчатой бумаги. Закрасим некоторое непустое множество клеток в черный цвет. Теперь мы хотим закрасить минимальное количество клеток, так, чтобы множество черных клеток стало выпуклым.

Напомним, что геометрическая фигура Ф называется выпуклой, если для любых точек A из Ф и B из Ф с вещественными координатами отрезок [AB] принадлежит Ф.

**Входные данные**

В первой строке входного файла INPUT.TXT содержатся два числа N и M ($1 \le N, M \le 100$) – размеры куска бумаги, куда попали все черные клетки. В каждой из следующих N строк содержится M символов «*» или «.». Символ «*» обозначает черную клетку, а «.» белую.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите выпуклое множество, содержащее минимальное количество дополнительно покрашенных черных клеток, в ровно N строках по M символов «*» или «.» в каждой.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 2 4<br>..*.<br>.**. | .**.<br>.**. |
| 2 | 4 3<br>.*.<br>.*.<br>.*.<br>.*. | .*.<br>.*.<br>.*.<br>.*. |

## 085. Единичный НОД

*(Время: 1 сек. Память: 16 Мб Сложность: 23%)*

Заданы два натуральных числа в десятичной системе счисления, состоящие из единиц. В первом числе ровно N единиц, а во втором их ровно M. Требуется найти НОД этих чисел.

Напомним, что НОД (наибольший общий делитель) двух чисел a и b – это такое максимальное число c, что b делится на c и a делится на c.

**Входные данные**

В единственной строке входного файла INPUT.TXT записаны два целых числа N и M ($1 \le N, M \le 2000$).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите ответ без ведущих нулей.

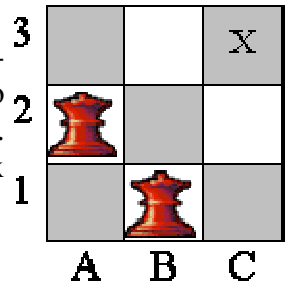
**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 1 1 | 1 |
| 2 | 1 2 | 1 |

## 086. Головоломка про ферзей

*(Время: 1 сек. Память: 16 Мб Сложность: 15%)*

Вероятно, что многие из вас играли в шахматы. Поэтому вы знаете, что ферзь может двигаться как по диагоналям, так и по горизонталям.




Вася недавно начал заниматься шахматами и где-то прочел головоломку, в которой нужно было расставить максимальное количество ферзей на доске 8x8 так, чтобы хотя бы одно поле оказалось небитым. Эта задача легко решается для доски 3x3, т.к. понятно, что более двух ферзей расставить таким образом на ней невозможно.

Помогите Васе решить эту задачу для доски NxN.

**Входные данные**

В единственной строке входного файла INPUT.TXT записано натуральное число N – размеры шахматной доски NxN ($1 \le N \le 100$).

**Выходные данные**

В единственную строку выходного файла OUTPUT.TXT нужно вывести максимальное количество ферзей, которых можно расставить на шахматной доске NxN так, чтобы одна клетка оставалась небитой.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3 | 2 |

## 087. Строки - 2

*(Время: 1 сек. Память: 16 Мб Сложность: 32%)*

Задано множество строк $S = \{s_1, s_2, s_3, \dots, s_n\}$. Необходимо найти количество строк $s_i$ из S, представимых в виде конкатенации двух строк $s_j$ и $s_k$ из S ($s_i = s_j s_k$, j и k при этом могут совпадать).

**Входные данные**

Входной файл INPUT.TXT содержит множество S – по одному элементу на строке. i-ая строка входного файла содержит $s_i$. Последняя строка входного файла содержит строку ENDOFINPUT. Она обозначает конец входных данных и не входит в множество S.

Все $s_i$ состоят только из маленьких букв латинского алфавита и имеют длину не более 100 символов. Во входном файле не более 239 строк.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите ответ без ведущих нулей.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | aa<br>aaaa<br>ab<br>abaa<br>ENDOFINPUT | 2 |
| 2 | abc<br>bcd<br>def<br>ENDOFINPUT | 0 |

## 088. Судоку

*(Время: 1 сек. Память: 16 Мб Сложность: 27%)*

Судоку размера n называется квадрат со стороной $n^2$, разделенный на $n^2$ средних квадратов со стороной n, каждый из которых разделен на $n^2$ маленьких квадратов. В каждом маленьком квадрате записано число от 1 до $n^2$.

Судоку называется правильным, если в каждом столбце, каждой строке и каждом среднем квадрате встречаются все числа от 1 до $n^2$.

Недавно Вася нарисовал Судоку размера n. Ваша задача – помочь ему определить правильный ли он.

**Входные данные**

В первой строке входного файла INPUT.TXT содержится число n ($1 \le n \le 10$). В следующих $n^2$ строках содержится по $n^2$ чисел, задающих нарисованный Васей Судоку.

Все числа во входном файле целые и не превосходят 100 по модулю.

**Выходные данные**

Если Судоку правильный, то выведите в выходной файл OUTPUT.TXT слово «Correct», иначе выведите «Incorrect».

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3<br>1 3 2 5 4 6 9 8 7<br>4 6 5 8 7 9 3 2 1<br>7 9 8 2 1 3 6 5 4<br>9 2 1 4 3 5 8 7 6<br>3 5 4 7 6 8 2 1 9<br>6 8 7 1 9 2 5 4 3<br>5 7 6 9 8 1 4 3 2<br>2 4 3 6 5 7 1 9 8<br>8 1 9 3 2 4 7 6 5 | Correct |
| 2 | 1<br>10 | Incorrect |

**Разбор**

Эта задача сводится к задаче, в которой необходимо определить: является ли представленный набор k чисел таковым, что он состоит из множества разных чисел от 1 до k (в нашем случае $k=n^2$). С одной стороны, можно было бы отсортировать этот набор чисел и проверить: в полученном массиве равно ли значение элемента его номеру. Этот метод прошел бы в силу того, что ограничения невелики, однако, этот способ неэффективен и сложнее по реализации, чем следующий. Мы можем определить некоторый массив d[1..k] (в нашем случае лучше описать такой массив из 100 элементов, независимо от n), заполнив его предварительно нулями. В процессе перебора всех чисел представленного набора элементов для каждого числа x будем выполнять присваивание d[x]=1, которое обеспечит факт присутствия числа x в списке. По окончании этой операции достаточно будет проверить все элементы массива d от 1 до k: если встретился хотя бы 1 ноль, то условие не выполнено, иначе все верно.

Описанный выше метод следует сначала применить для элементов каждой строки, далее для элементов каждого столбца, потом для каждого подмассива размерности n x n. Последняя операция, вероятно, более сложна, поэтому приведем примерный код перебора всех таких подмассивов:

```
for i=1..n{
  for j=1..n{
    d[1..100]={0..0};
    for y=(i-1)*n+1 .. i*n{
      for x=(j-1)*n+1 .. j*n{
        d[a[y][x]]=1;
      }
    }
    Check(d);
  }
}
```

## 089. Быстрый поезд

*(Время: 1 сек. Память: 16 Мб Сложность: 25%)*

Между двумя крупнейшими городами нашей страны Санкт-Петербургом и Москвой ежедневно совершают рейсы n поездов. Для каждого поезда известно его время отправления из Санкт-Петербурга и время прибытия в Москву.

Найдите самый быстрый поезд и его скорость в предположении, что длина железной дороги между Санкт-Петербургом и Москвой равна 650 км.

**Входные данные**

Первая строка входного файла INPUT.TXT содержит целое число n ($1 \le n \le 100$). Каждая из последующих n строк описывает ровно один поезд.

Описание поезда состоит из его названия, времени отправления и времени прибытия. Название поезда – строка длиной не более 50 символов, заключенная в кавычки. Она может содержать буквы латинского алфавита, пробелы, цифры, символы тире («–») и подчеркивания («_»). Времена отправления и прибытия заданы в формате ЧЧ:ММ. Строчные и заглавные буквы в названиях поездов различаются.

Время в пути для каждого из поездов составляет хотя бы одну минуту и не превышает 24 часов.

Гарантируется, что самый быстрый поезд определяется единственным образом.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите название самого быстрого поезда и его скорость. Скорость выводите в километрах в час и округляйте до целых. Следуйте формату вывода, приведенному в примерах.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3<br>"ER-200" 06:43 10:40<br>"Red Arrow" 23:55 07:55<br>"Express" 23:59 08:00 | The fastest train is "ER-200".<br>It's speed is 165 km/h, approximately. |
| 2 | 3<br>"Train1" 00:00 00:00<br>"Train2" 00:00 00:01<br>"Train3" 00:01 00:01 | The fastest train is "Train2".<br>It's speed is 39000 km/h, approximately. |
| 3 | 2<br>"Slow Train 1" 10:00 09:59<br>"Slow Train 2" 10:00 10:00 | The fastest train is "Slow Train 1".<br>It's speed is 27 km/h, approximately. |

## 090. Треугольные страны

*(Время: 1 сек. Память: 16 Мб Сложность: 37%)*

Эта история происходила на одной плоской планете. С незапамятных времен на ней существовал город N, находящийся в точке $x_n$,$y_n$. Кроме этого, в разное время на этой же планете существовали страны, каждая из которых имела форму треугольника.

Теперь перед историками встала серьезная задача – по имеющимся у них данным о треугольных странах определить, в какие страны мог входить город N. Город мог входить в страну, если он находится строго внутри нее.

**Входные данные**

Первая строка входного файла содержит два числа: $x_n$ и $y_n$ – координаты города N. Вторая строка входного файла содержит количество k треугольных стран ($1 \le k \le 1000$). Последующие k строк каждая описывают одну треугольную страну. Описание треугольной страны состоит из шести целых чисел $x_1$, $y_1$, $x_2$, $y_2$, $x_3$, $y_3$, где ($x_1$, $y_1$), ($x_2$, $y_2$), ($x_3$, $y_3$) – координаты вершин этой страны.

Гарантируется, что все страны имеют ненулевую площадь. Все координаты не превосходят 10000 по абсолютной величине.

**Выходные данные**

В первой строке выходного файла выведите количество стран, в которые мог входить город N. Во второй строке выведите через пробел номера этих стран в возрастающем порядке. Страны нумеруются с единицы в том порядке, в каком они заданы во входном файле.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 0 1<br>2<br>-2 0 2 0 0 2<br>-3 0 3 0 0 3 | 2<br>1 2 |
| 2 | 0 2<br>2<br>-2 0 2 0 0 2<br>-3 0 3 0 0 3 | 1<br>2 |

## 091. Две последовательности

*(Время: 1 сек. Память: 16 Мб Сложность: 29%)*

Определим последовательности $a_n$ и $b_n$ следующим образом: $a_1 = 2$, $a_2 = 3$, $a_3 = 4$, $a_4 = 7$, $a_n = b_{n-1} + b_{n-3}$, $n > 4$, $b_n$ – последовательность чисел, не входящих в $a_n$, записанных в возрастающем порядке.

Таким образом, последовательность $a_n$ будет выглядеть следующим образом: 2, 3, 4, 7, 13, 15,..., а последовательность $b_n$ – 1, 5, 6, 8, 9, 10,....

Ваша задача состоит в том, чтобы найти $a_n$ и $b_n$.

**Входные данные**

Входной файл содержит целое число n ($1 \le n \le 10000$).

**Выходные данные**

В первой строке выходного файла выведите $a_n$, во второй – $b_n$.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 4 | 7<br>8 |
| 2 | 10 | 25<br>16 |
| 3 | 6578 | 19731<br>9868 |

## 092. Журавлики

*(Время: 1 сек. Память: 16 Мб Сложность: 7%)*

Петя, Катя и Сережа делают из бумаги журавликов. Вместе они сделали S журавликов. Сколько журавликов сделал каждый ребенок, если известно, что Петя и Сережа сделали одинаковое количество журавликов, а Катя сделала в два раза больше журавликов, чем Петя и Сережа вместе?

**Входные данные**

В единственной строке входного файла INPUT.TXT записано одно натуральное число S – общее количество сделанных журавликов ($S < 10^6$).

**Выходные данные**

В единственную строку выходного файла OUTPUT.TXT нужно вывести три числа, разделенных пробелами – количество журавликов, которые сделал каждый ребенок (Петя, Катя и Сережа).

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 6 | 1 4 1 |
| 2 | 24 | 4 16 4 |
| 3 | 60 | 10 40 10 |

**Разбор**

В этой задаче нужно решить задачу по математике для 4-го класса с составлением уравнения. Пусть x – число журавликов, которые сделали Петя и Сережа, тогда Катя сделала 4x журавликов, ну а поскольку журавликов всего S, то x+x+4x=S, а значит x=S/6. Откуда так же получаем что Катя сделала 2*S/3 журавликов.

Следует отметить на ограничения на входные данные. Здесь известно, что S менее миллиона, поэтому можно для хранения использовать 4-байтовый целый тип. Особенно внимательно к этому следует отнестись при программировании в среде DOS. Ведь там тип int (integer) занимает 2 байта и задача может не пройти все тесты. Так же заметим, что нигде не сказано, что S делится на 6, но в этой задаче это так, хотя вполне могло бы быть и иначе...

## 093. Боги

*(Время: 1 сек. Память: 16 Мб Сложность: 26%)*

Археологами найден набор древних копий старинных манускриптов с мифами – различными историями о древних богах. К несчастью, переписчики этих манускриптов не отличались особой грамотностью и умудрились в каждом имени сделать ровно по одной орфографической ошибке – т.е. ровно одну из букв божественного имени заменили какой-то другой буквой. Археологи смогли составить список правильных написаний имен богов, так же им удалось выписать из манускриптов все имена собственные. Однако сопоставлять два списка – свыше их сил. Помогите им в этом!

**Входные данные**

Первая строка входного файла INPUT.TXT содержит число N – количество имен богов в списке. Следующие N строк – имена богов. Далее идет строка, содержащая число M – количество «подозрительных» слов, выписанных из манускриптов. Следующие M строк – «подозрительные» слова. Каждое из имен богов и «подозрительных» слов – последовательность из K заглавных букв английского алфавита ($1 \le N, M, K \le 30$).

**Выходные данные**

В выходной файл OUTPUT.TXT выводится N чисел – для каждого божьего имени выводится число «подозрительных» слов, которые являются именем данного бога с одной ошибкой.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3<br>ZEUS<br>POSEIDON<br>AFINA<br>4<br>ZEVS<br>POSEYDON<br>AVYNA<br>ZERS | 2 1 0 |

## 094. Принц и дракон

*(Время: 1 сек. Память: 16 Мб Сложность: 22%)*

Волшебник Мерлин продает волшебные мечи принцам, желающим убить дракона. Основная характеристика меча – число драконьих голов, которые он срубает за удар. Основная характеристика дракона – число голов, которые он может отрастить за сеанс регенерации. Бои принцев с драконами всегда протекают одинаково – принц атакует, и прячется за щитом;

дракон атакует огненным дыханием и регенерирует; так продолжается до тех пор, пока после очередного удара у дракона не кончатся головы. Ясно, впрочем, что не каждым мечом можно победить каждого дракона. Заказ, поступающий Мерлину, всегда содержит число голов дракона и скорость его регенерации. Подсчитайте по известной атакующей силе меча, сможет ли принц убить такого дракона таким мечом и, если да, то сколько ударов потребуется.

**Входные данные**

Единственная строка входного файла INPUT.TXT содержит число N – число голов, которые меч срубает одним ударом. Далее идет число M – число голов дракона. За ним идет K – число голов, которые дракон регенерирует за раз ($1 \le N, M, K \le 10^5$). Все числа разделены пробелом.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите число ударов, которые необходимо нанести принцу, чтобы убить дракона, если это возможно. Если таким мечом убить дракона нельзя, то следует вывести «NO».

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3 6 2 | 4 |
| 2 | 4 4 5 | 1 |
| 3 | 5 10 6 | NO |

## 095. Нумеролог

*(Время: 1 сек. Память: 16 Мб Сложность: 24%)*

Чтобы предсказать судьбу человека, нумеролог берет время жизни человека в секундах, затем складывает все цифры этого числа. Если полученное число состоит более чем из одной цифры, операция повторяется, пока в числе не останется одна цифра. Затем по полученной цифре и числу операций, необходимых для преобразования числа в цифру нумеролог предсказывает судьбу человека. Нумеролог плохо умеет считать, а числа, с которыми он работает, могут быть очень большими. Напишите программу, которая бы делала все расчеты за него.

**Входные данные**

Входного файла INPUT.TXT содержит число N – время жизни человека в секундах ($1 \le N \le 10^{100}$).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите два числа через пробел: полученную цифру из числа N и число преобразований.

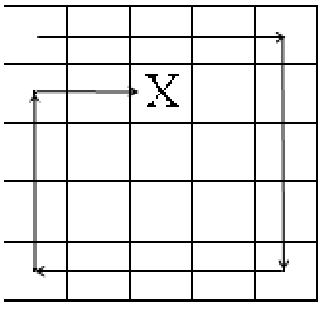
**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 1 | 1 0 |
| 2 | 10 | 1 1 |
| 3 | 99 | 9 2 |

## 096. Винни-Пух

*(Время: 1 сек. Память: 16 Мб Сложность: 41%)*

Вини-Пух стоит на прямоугольном поле размером N*M клеток. В каждой клетке растет по одной ягоде. В начальный момент времени он стоит на левой верхней клетке. Он начинает собирать ягоды по верхнему краю поля. Если он доходит до края поля или до пустой клетки, он поворачивается на 90 градусов вправо и продолжает собирать ягоды. Но дойдя до заданной клетки Вини вспоминает, что его ждет Пятачок, и он уходит с поля.



**Входные данные**

В первой строке входного файла INPUT.TXT стоят размеры поляны N и M ($0 < N, M \le 100$) – высота и ширина, во второй числа Y и X ($0 < Y \le N$, $0 < X \le M$) – номера строки и столбца клетки, дойдя до которой Вини-Пух прекращает собирать ягоды.

**Выходные данные**

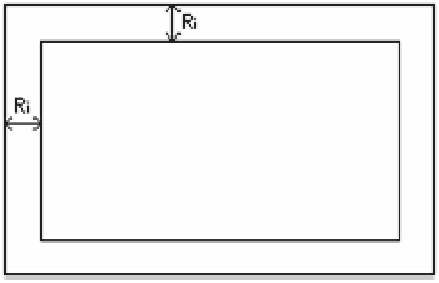
В выходной файл OUTPUT.TXT выведите число ягод, которые соберет Пух.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 1 1<br>1 1 | 1 |
| 2 | 3 3<br>2 3 | 4 |
| 3 | 5 5<br>2 3 | 18 |

## 097. Заповедники

*(Время: 1 сек. Память: 16 Мб Сложность: 60%)*




В райской долине расположены N заповедников, имеющих форму прямоугольников. Однажды на собрании директоров было принято решение об увеличении площадей заповедников. Для этого директор каждого заповедника выбрал Ri – количество метров, на которое он хочет увеличить зону своего заповедника, смотрите рисунок. Однако после подписания соглашения выяснилось, что некоторые заповедники имеют общие земли. Такие заповедники было решено объединить в один, если объединенный заповедник пересекался с еще каким-нибудь заповедником их опять объединяли и так до тех пор пока не остались заповедник(и) не имеющие общих земель.

Ваша задача посчитать, сколько заповедников стало в долине после объединения.

**Входные данные**

Первая строка входного файла INPUT.TXT содержит число N ($1 \le N \le 100$) – количество заповедников. Далее идет N строк содержащих по пять целых чисел $x_1$, $y_1$, $x_2$, $y_2$, R. ($x_1$, $y_1$) и ($x_2$, $y_2$) – координаты противоположных вершин заповедника в метрах ($-10^4 \le x_1, y_1, x_2, y_2 \le 10^4$). Стороны заповедников параллельны осям координат. Заповедники, имеющие общую границу, считаются пересекающимися. R ($0 \le R \le 10^4$) – расстояние, на которое отодвигается граница заповедника.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите одно натуральное число – количество оставшихся заповедников после объединения.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3 1 6 4 1<br>1 -2 2 -3 1<br>-2 -2 -1 -3 2 | 2 |

## 098. Игра в числа

*(Время: 1 сек. Память: 16 Мб Сложность: 28%)*

Игра в числа ведётся на одномерном массиве целых положительных чисел. Перед началом, жеребьёвкой определяется, кто будет ходить первым (первый игрок), а кто – вторым (второй игрок). Процесс игры состоит в том, что игроки по очереди (сначала первый игрок, затем второй, следом опять первый и так далее) вычёркивают числа из массива. Вычеркнуть

можно только число, находящееся в конце или начале оставшегося массива. При этом всегда вычёркивается максимальное число из этих двух. Если первое и последнее числа массива равны, то вычёркивается первое. Игра продолжается до того момента, пока не будут вычеркнуты все числа. Каждое вычеркнутое число идёт в актив тому игроку, который его вычеркнул. После окончания игры каждый игрок суммирует вычеркнутые им числа. Победителем объявляется тот, кто наберет больше очков.

Некоторые игроки поняли, что результат не зависит от стратегии игры, и решили попросить Вас написать программу для получения результата.

**Входные данные**

В первой строке входного файла INPUT.TXT находится одно целое число N – количество чисел в массиве ($1 \le N \le 10^4$). Во второй строке находятся N целых положительных чисел из диапазона [1, 32000], разделённых пробелом.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите два числа, разделенные двоеточием. Первое число – количество очков, набираемых первым игроком при игре на этом массиве, второе число – для второго.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 5<br>4 4 1 5 4 | 9:9 |
| 2 | 1<br>1234 | 1234:0 |

## 099. Лабиринт

*(Время: 1 сек. Память: 16 Мб Сложность: 57%)*

Открыв глаза, Принц Персии обнаружил, что находится на верхнем уровне подземного лабиринта Джаффара. Лабиринт состоит из h уровней, расположенных строго друг под другом. Каждый уровень представляет собой прямоугольную площадку, разбитую на m x n участков. На некоторых участках стоят колонны, поддерживающие потолок, на такие участки Принц заходить не может.

Принц может перемещаться с одного участка на другой свободный участок того же уровня, так же он может проломить пол под собой и оказаться уровнем ниже (на самом нижнем уровне пол проломить нельзя). Любое перемещение занимает у Принца 5 секунд.

На одном из участков нижнего уровня Принца ждет Принцесса. Помогите Принцу найти Принцессу, потратив на это как можно меньше времени.

**Входные данные**

В первой строке входного файла INPUT.TXT содержатся натуральные числа h, m и n – высота и горизонтальные размеры лабиринта ($2 \le h, m, n \le 50$). Далее во входном файле приведены h блоков, описывающих уровни лабиринта в порядке от верхнего к нижнему. Каждый блок содержит m строк, по n символов в каждой: «.» обозначает свободный участок, «o» обозначает участок с колонной, «1» обозначает свободный участок, в котором оказался Принц в начале своего путешествия, «2» обозначает свободный участок, на котором томится Принцесса. Символы «1» и «2» встречаются во входном файле ровно по одному разу: символ «1» – в описании самого верхнего уровня, а символ «2» – в описании самого нижнего. Соседние блоки разделены одной пустой строкой.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите минимальное время в секундах, необходимое Принцу, чтобы найти Принцессу. Поскольку Добро всегда побеждает Зло, гарантируется, что Принц может это сделать.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3 3 3<br>1..<br>oo.<br>...<br><br>Ooo<br>..o<br>.oo<br><br>Ooo<br>o..<br>o.2 | 60 |

## 100. Счастливые билеты

*(Время: 1 сек. Память: 16 Мб Сложность: 86%)*

Требуется вычислить количество N - значных счастливых билетов. Напомним, что билет называется счастливым, если сумма первой половины его цифр равна сумме другой его половины. Например, билет 564159 счастливый, т.к. 5+6+4=1+5+9.

**Входные данные**

В единственной строке входного файла INPUT.TXT записано натуральное четное число N ($N \le 100$) – количество цифр в билете.

**Выходные данные**

В единственную строку выходного файла OUTPUT.TXT нужно вывести одно целое число – количество N-значных счастливых билетов.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 4 | 670 |
| 2 | 6 | 55252 |
| 3 | 12 | 39581170420 |

## 101. Магараджа

*(Время: 1 сек. Память: 16 Мб Сложность: 70%)*

Магараджа – это шахматная фигура, сочетающая возможности ферзя и коня. Таким образом, магараджа может ходить и бить на любое количество клеток по диагонали, горизонтали и вертикали (т.е. как ферзь), а также либо на две клетки по горизонтали и на одну по вертикали, либо на одну по горизонтали и на две по вертикали (как конь).

Ваша задача – найти число способов расставить на доске N на N ровно K магараджей так, чтобы они не били друг друга.

**Входные данные**

Входной файл INPUT.TXT содержит два целых числа: N и K ($1 < K \le N \le 10$).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите ответ на задачу.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3 1 | 9 |
| 2 | 4 2 | 20 |
| 3 | 5 3 | 48 |

## 102. Треугольник

*(Время: 1 сек. Память: 16 Мб Сложность: 32%)*

В декартовой системе координат на плоскости заданы координаты вершин треугольника и еще одной точки. Требуется написать программу, определяющую, принадлежит ли эта точка треугольнику.

**Входные данные**

В четырех строках входного файла INPUT.TXT находятся пары целых чисел – координаты точек. Числа в первых трех строках – это координаты вершин треугольника ($x_1$, $y_1$), ($x_2$, $y_2$), ($x_3$, $y_3$), в четвертой строке – координаты тестируемой точки ($x_4$, $y_4$). Все координаты не превышают 10000 по абсолютной величине.

**Выходные данные**

В выходной файл OUTPUT.TXT необходимо вывести слово «In», если точка находится внутри треугольника и «Out» в противном случае.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 0 0<br>100 0<br>0 100<br>100 100 | Out |
| 2 | 0 0<br>100 0<br>0 100<br>10 10 | In |
| 3 | 0 0<br>100 0<br>0 100<br>50 50 | In |
| 4 | 0 0<br>100 0<br>0 100<br>0 0 | In |

## 103. Снова A+B

*(Время: 1 сек. Память: 16 Мб Сложность: 35%)*

Требуется сложить два целых числа A и B.

**Входные данные**

В единственной строке входного файла INPUT.TXT записано два неотрицательных целых числа через пробел, не превышающих $10^{100}$.

**Выходные данные**

В единственную строку выходного файла OUTPUT.TXT нужно вывести одно целое число – сумму чисел A и B.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 2 3 | 5 |

**Разбор**

Сложение двух длинных чисел выполняется с использованием тех же принципов, что и при поразрядном сложении вручную на листе бумаги. Считаем сначала числа в массивы a и b, после чего осуществим сложение таким образом, что результат суммы запишется в массив a. В переменной c будем хранить перенос на следующий разряд.

В простейшем случае решение поставленной задачи представимо следующим алгоритмом:

```
const maxsize=101;
int a[maxsize], b[maxsize];

readlong(a);
readlong(b);

m=max(a[0],b[0]);

c=0;
for i=1..m{
  c = c+a[i]+b[i];
  a[i] = c mod 10;
  c = c div 10;
}
a[0]=m;
if(c>0){
  m=m+1;
  a[m] = c;
}

writelong(a);
```

## 104. Шаблон

*(Время: 1 сек. Память: 16 Мб Сложность: 65%)*

Будем рассматривать слова из больших латинских букв и шаблоны, состоящие из больших латинских букв и символов «?» и «*». Говорят, что слово подходит под шаблон, если в шаблоне можно заменить каждый символ «?» на большую латинскую букву, а каждый символ «*» – на последовательность (возможно, пустую) больших латинских букв, так, чтобы получилось требуемое слово. Требуется написать программу, определяющую, подходит ли слово под шаблон.

**Входные данные**

В первых двух строках входного файла INPUT.TXT записаны шаблон и слово: в одной строке записан шаблон – последовательность больших латинских букв, «?» и «*», в другой – слово, состоящее только из больших латинских букв. Обе строки входного файла не превышают 255 символов.

**Выходные данные**

В выходной файл OUTPUT.TXT необходимо вывести слово «YES», если слово подходит под шаблон и «NO» в противном случае.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | ABBCDA<br>A*CDA | YES |

## 105. Раскопки

*(Время: 1 сек. Память: 16 Мб Сложность: 76%)*

Во время недавних раскопок на Марсе были обнаружены листы бумаги с таинственными символами на них. После долгих исследований учёные пришли к выводу, что надписи на них на самом деле могли быть обычными числовыми равенствами. Кроме того, из других источников было получено веское доказательство того, что марсиане знали только три операции – сложение, умножение и вычитание (марсиане никогда не использовали «унарный минус»: вместо «-5» они писали «0-5»). Также ученые доказали, что марсиане не наделяли операции разным приоритетом, а просто вычисляли выражения (если в них не было скобок) слева направо: например, 3+3*5 у них равнялось 30, а не 18. К сожалению, символы арифметических действий стерлись. Например, если была запись «18=7 (5 3) 2», то возможно восстановить эту запись как «18=7+(5-3)*2». Требуется написать программу, находящую требуемую расстановку знаков или сообщающую, что таковой не существует.

**Входные данные**

Первая строка входного файла INPUT.TXT состоит из натурального числа, не превосходящего 230, знака равенства, и последовательности натуральных чисел (не более десяти), произведение которых также не превосходит 230. Некоторые группы чисел (одно или более) могут быть окружены скобками. Длина входной строки не будет превосходить 80 символов, и других ограничений на количество и вложенность скобок нет. Между двумя соседними числами, не разделенными скобками, всегда будет хотя бы один пробел, во всех остальных местах может быть любое (в том числе и 0) число пробелов (естественно, внутри числа пробелов нет).

**Выходные данные**

В выходной файл OUTPUT.TXT необходимо вывести одну строку, содержащую полученное равенство (т.е., исходное равенство со вставленными знаками арифметических действий без лишних пробелов). В случае если требуемая расстановка знаков невозможна, вывести строку, состоящую из единственного числа «-1». Выходная строка не должна содержать пробелов.

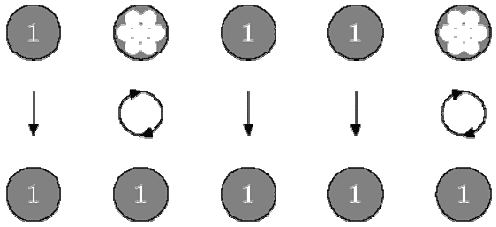
**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | `18=7 (5 3) 2` | `18=7+(5-3)*2` |
| 2 | `5= 3 3` | `-1` |

## 106. Монетки

*(Время: 1 сек. Память: 16 Мб Сложность: 8%)*

На столе лежат n монеток. Некоторые из них лежат вверх решкой, а некоторые – гербом. Определите минимальное число монеток, которые нужно перевернуть, чтобы все монетки были повернуты вверх одной и той же стороной.



**Входные данные**

В первой строке входного файла INPUT.TXT записано натуральное число N ($1 \le N \le 100$) – число монеток. В каждой из последующих N строк содержится одно целое число: 1, если монетка лежит решкой вверх и 0, если вверх гербом.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите минимальное количество монет, которые нужно перевернуть.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | `5`<br>`1`<br>`0`<br>`1`<br>`1`<br>`0` | `2` |

**Разбор**

В этой задаче нужно вычислить число монеток, которые лежат гербом и число монеток, лежащих решкой. После чего следует вывести наименьшее значение. Для этого не обязательно использовать массив, т.к. достаточно сначала считать в некоторую переменную n общее число монеток, а потом в цикле проводить чтение i-й монетки и в зависимости от ее значения увеличивать одну из двух переменных на единицу.

Приведем алгоритм решения этой задачи:

```
read(n);
s0=s1=0;
for i=1..n {
  read(x);
  if(x==0) s0++; else s1++;
}
write(min(s0,s1));
```

## 107. Красивые номера

*(Время: 1 сек. Память: 16 Мб Сложность: 62%)*

Вы, наверное, замечали, что многие компании используют для рекламы «красивые» номера телефонов, которые удобны для запоминания потенциальными клиентами. Но что делать, если номер вашей компании ничем не примечателен? Можно присмотреться к нему повнимательнее, а вдруг, если перегруппировать цифры номера некоторым образом, номер станет намного красивее? Например, если у вашей компании номер 872-73-33, то его можно сделать красивее, если перегруппировать цифры так: 8727-333.

Введем следующую оценку красоты разбиения номера. Будем разбивать номер дефисами на группы размером от 2 до 4 цифр. Теперь красотой разбиения назовем сумму баллов, которые приносит каждая группа. Эти баллы будем считать, пользуясь приведенной таблицей.

| Шаблон группы | Баллы |
|---|---|
| aa | 2 |
| aba | 2 |
| aab, abb | 2 |
| aaa | 3 |
| abac, baca | 2 |
| abab | 3 |
| aabb | 3 |
| abba | 4 |
| baaa, abaa, aaba, aaab | 3 |
| aaaa | 5 |

В этой таблице символами «a», «b», «c» обозначены различные цифры. Например, под шаблон «aab» подходят группы «223», «667», но не подходят «123» и «888». Пользуясь предложенной оценкой, найдите наиболее красивое разбиение заданного номера.

**Входные данные**

Входной файл INPUT.TXT содержит одну строку из 7 цифр – заданный телефонный номер.

**Выходные данные**

Выведите в первой строке выходного файла OUTPUT.TXT наиболее красивое разбиение номера, а во второй – величину его красоты. Если разбиений с максимальной величиной красоты несколько, выведите в выходной файл любое из этих разбиений.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 8727333 | 8727-333<br>5 |
| 2 | 8827291 | 88-272-91<br>4 |

## 108. Неглухой телефон

*(Время: 1 сек. Память: 16 Мб Сложность: 1%)*

Возможно, что Вы когда то играли в игру «Глухой телефон», либо слышали о ней. В этой игре участникам приходится передавать информацию друг другу различными способами: словесно, образно, бывает даже приходится писать левой рукой текст, который другой

участник команды должен будет прочитать. Так же известно, что практически никогда передаваемая информация не доходит до конечного адресата. Обозначим за $F_i(x)$ функцию, которая преобразует текст передаваемой информации x в ту, которую получит участник i+1 от участника i. Тогда последний n-й участник получит данные y, которые будут выражаться следующей формулой:

$$y = F_{n-1}(F_{n-2}(\ldots F_2(F_1(x)))).$$

Но Вам необходимо исключить какие-либо внешние факторы, которые могут исказить исходную информацию и Вы должны реализовать программу «неглухой телефон», которая сможет безошибочно доставлять исходные данные, т.е. в нашем случае функция $F_i(x) = x$ для всех i от 1 до n-1.

**Входные данные**

В единственной строке входного файла INPUT.TXT записано натуральное число от 1 до 100.

**Выходные данные**

В выходной файл OUTPUT.TXT нужно вывести в точности то же число, которое задано во входном файле.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 5 | 5 |

**Разбор**

Эта задача скорее является пародией на класс легких задач со сложной формулировкой, т.к. уж слишком она простая. Дело в том, что часто на олимпиадах попадается одна задача, имеющая легкое решение, но пугающая участников своей сложной формулировкой. Так и здесь: в задании имеется масса текста, включающего какие-то рекуррентные математические формулы, наводящие на мысль о том, что задача может быть серьезной. А на самом то деле оказывается, что нужно считать число из input.txt и вывести его же в output.txt . Разве может быть еще что-то легче? (на самом деле может :) – вывести число 1 не зависимо от того, что в файле input.txt, но это уже слишком...)

Поэтому иногда полезно, увидев большой текст задания обратить внимание на то, что записано в разделах входных и выходных данных, иногда просто этого оказывается достаточно для решения задачи. Считаем, что теперь Вы без труда сможете справиться с этой, пожалуй, самой легкой в этой системе задачей. Успехов!

## 109. A / B

*(Время: 1 сек. Память: 16 Мб Сложность: 43%)*

Требуется получить точное значение частного A/B для двух натуральных чисел A и B.

**Входные данные**

В единственной строке входного файла INPUT.TXT записано частное двух натуральных чисел, не превышающих 1000. Числа разделены символом «/» без лишних пробелов.

**Выходные данные**

В выходной файл OUTPUT.TXT нужно вывести точное значение A/B без лишних точек, нулей и пробелов. В случае присутствия бесконечной записи числа следует вывести период в скобках.

Например, неправильно выведены числа: 08.92, 3.20, 120.6(6), 0.(33), 5.(0), 2. , .3, 0.33(03) . Их следует выводить как 8.92, 3.2, 120.(6), 0.(3), 5, 2, 0.3, 0.3(30) .

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 10/7 | 1.(428571) |
| 2 | 1/3 | 0.(3) |
| 3 | 100/25 | 4 |

## 110. Красивые числа

*(Время: 1 сек. Память: 16 Мб Сложность: 78%)*

Саша считает красивыми числа, десятичная запись которых не содержит других цифр, кроме 0 и k ($1 \le k \le 9$). Например, если k = 2, то такими числами будут 2, 20, 22, 2002 и т.п. Остальные числа Саше не нравятся, поэтому он представляет их в виде суммы красивых чисел. Например, если k = 3, то число 69 можно представить так: 69 = 33 + 30 + 3 + 3.

Однако не любое натуральное число можно разложить в сумму красивых целых чисел. Например, при k = 5 число 6 нельзя представить в таком виде. Но если использовать красивые десятичные дроби, то это можно сделать: 6 = 5.5 + 0.5.

Недавно Саша изучил периодические десятичные дроби и начал использовать и их в качестве слагаемых. Например, если k = 3, то число 43 можно разложить так: 43 = 33.(3) + 3.(3) + 3 + 3.(3).

Оказывается, любое натуральное число можно представить в виде суммы положительных красивых чисел. Но такое разложение не единственно – например, число 69 можно также представить и как 69 = 33 + 33 + 3. Сашу заинтересовало, какое минимальное количество слагаемых требуется для представления числа n в виде суммы красивых чисел.

Требуется написать программу, которая для заданных чисел n и k находит разложение числа n в сумму положительных красивых чисел с минимальным количеством слагаемых.

**Входные данные**

Во входном файле INPUT.TXT записаны два натуральных числа n и k ($1 \le n \le 10^9$, $1 \le k \le 9$).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите разложение числа n в сумму положительных чисел, содержащих только цифры 0 и k, количество слагаемых в котором минимально. Разложение должно быть представлено в виде: $n=a_1+a_2+...+a_m$. Слагаемые $a_1$, $a_2$, ..., $a_m$ должны быть выведены без ведущих нулей, без лишних нулей в конце дробной части. Запись каждого слагаемого должна быть такой, что длины периода и предпериода дробной части имеют минимально возможную длину.

Например, неправильно выведены числа: 07.7; 2.20; 55.5(5); 0.(66); 7.(0); 7. ; .5; 0.33(03). Их следует выводить так: 7.7; 2.2; 55.(5); 0.(6); 7; 7; 0.5; 0.3(30). Предпериод и период каждого из выведенных чисел должны состоять не более чем из 100 цифр. Гарантируется, что хотя бы одно такое решение существует. Числа ряда следует выводит в порядке неубывания. Выходной файл не должен содержать пробелов.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | `69 3` | `69=3+33+33` |
| 2 | `6 5` | `6=0.5+5.5` |
| 3 | `10 9` | `10=9.(9)` |

## 111. Игра «Пуговицы»

*(Время: 1 сек. Память: 16 Мб Сложность: 28%)*

Правила игры очень просты. Перед двумя играющими находится кучка из K пуговиц. Играющие по очереди берут пуговицы из кучки, причем за один ход каждый из них может взять от 1 до L пуговиц. Выигрывает тот из спортсменов, которому удастся взять последнюю пуговицу.

Тот из игроков, которому по жребию выпадает делать первый ход, получает возможность собственноручно назначить число K. Тот из игроков, который будет ходить вторым, выбирает, в свою очередь, число L.

Вам необходимо определить наилучшую стратегию для участника, который ходит вторым.

**Входные данные**

Во входном файле INPUT.TXT записано одно натуральное число K ($1 \le K \le 10^8$) – общее количество пуговиц.

**Выходные данные**

В выходной файл OUTPUT.TXT необходимо вывести целое число L ($2 \le L < K$) – максимальное количество пуговиц, которое можно взять за один ход, обеспечивающее победу второму игроку. Если таких чисел несколько, то следует вывести наименьшее из них. Если таких чисел нет, то следует вывести число 0.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3 | 2 |
| 2 | 26 | 12 |
| 3 | 31 | 30 |

## 112. Армия

*(Время: 1 сек. Память: 16 Мб Сложность: 58%)*

Всем известно, что в армии без строевой подготовки и порядка дело не обходится и за этим там строго следят. Однажды утром сержант построил всех своих подчиненных в K рядов по N человек в каждом, но оказалось, что солдаты выстроились не по росту, и поэтому сержант решил их наказать. Солдаты должны были выстроиться по росту в каждом отдельном ряде так, что слева должны были стоять самые низкие, а справа самые высокие. Ну а поскольку в армии виноваты всегда слабые (низкие), то наказание было следующим: каждый солдат должен был отжаться столько раз, сколько солдат стоит от него слева выше его ростом.

Оказалось, что все солдаты были разного роста, и многим пришлось отжиматься достаточно много раз. Сержанту стало интересно: сколько же раз в общей сложности пришлось отжаться солдатам?

Помогите ему решить эту задачу!

**Входные данные**

В первой строке входного файла INPUT.TXT записаны два натуральных числа N и K ($2 \le N \le 10^4$, $1 \le K \le 20$) – число солдат в каждом ряде и число рядов. Следующие K строк файла содержат ровно N разных натуральных чисел от 1 до N – рост солдат. Первое число ряда – рост первого солдата (самого левого в ряду), второе – рост второго, и т.д.

**Выходные данные**

В выходной файл OUTPUT.TXT необходимо вывести общее количество отжиманий, которые должны были выполнить солдаты.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3 3<br>1 2 3<br>2 1 3<br>3 2 1 | 4 |
| 2 | 5 2<br>1 5 2 4 3<br>2 3 1 5 4 | 7 |

## 113. Фермер

*(Время: 1 сек. Память: 16 Мб Сложность: 46%)*

Фермер решил на своем квадратном участке земли вспахать пашню квадратной формы максимальной площади, т.к. он посчитал, что именно квадратная форма пашни наиболее удобна для обработки. Но на его участке присутствуют деревья и хозяйственные постройки, которые он никуда не хочет переносить, а так же иные места, не пригодные для пашни. Для удобства он составил квадратную карту местности NxN в форме матрицы и пометил нулями непригодные для пашни зоны, в остальные зоны он поставил единицу.

Необходимо помочь фермеру определить максимальную площадь пашни.

**Входные данные**

В первой строке входного файла INPUT.TXT записано единственное натуральное число N ($1 \le N \le 1000$) – длина стороны квадратного участка фермы. Далее, следует N строк, в каждой из которых находится последовательность (без пробелов) нулей и единиц, описывающих ферму.

**Выходные данные**

В выходной файл OUTPUT.TXT необходимо вывести максимально возможную площадь пашни.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 7<br>1101101<br>1111110<br>1011100<br>0011100<br>1000010<br>1100111<br>1001110 | 9 |

## 114. Без двух нулей подряд

*(Время: 1 сек. Память: 16 Мб Сложность: 37%)*

Требуется вычислить количество N-значных чисел в системе счисления с основанием K, таких, что их запись не содержит двух подряд идущих нулей.

**Входные данные**

Во входном файле INPUT.TXT записаны два натуральных числа N и K в десятичной системе счисления ($2 \le K \le 10$; $2 \le N$; $4 \le N+K \le 18$).

**Выходные данные**

В выходной файл OUTPUT.TXT необходимо вывести целое число в десятичной записи – ответ на задачу.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 2 10 | 90 |
| 2 | 4 2 | 5 |
| 3 | 6 3 | 328 |

## 115. Прямоугольник

*(Время: 1 сек. Память: 16 Мб Сложность: 42%)*

Задан целочисленный прямоугольный массив MxN. Необходимо определить прямоугольную область данного массива, сумма элементов которого максимальна.

**Входные данные**

В первой строке входного файла INPUT.TXT записаны два натуральных числа N и M ($1 \le N, M \le 100$) – количество строк и столбцов прямоугольной матрицы. Далее идут N строк по M чисел, записанных через пробел – элементы массива, целые числа, не превосходящие 100 по абсолютной величине.

**Выходные данные**

В выходной файл OUTPUT.TXT необходимо вывести целое число – сумму элементов найденного прямоугольного подмассива.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 2 3<br>5 0 9<br>1 2 7 | 24 |
| 2 | 4 5<br>-7 8 -1 0 -2<br>2 -9 2 4 -6<br>-7 0 6 8 1<br>4 -8 -1 0 -6 | 20 |

## 116. Фермер - 2

*(Время: 1 сек. Память: 16 Мб Сложность: 60%)*

После решения задачи с пашней земли, фермер хочет построить на этой земле как можно больший по площади сарай прямоугольной формы. Но на его участке есть деревья и хозяйственные постройки, которые он не хочет никуда переносить. Для простоты представим ферму прямоугольной сеткой размера MxN. Каждое из деревьев и построек размещается в одном или нескольких узлах сетки. Сарай должен быть построен на свободных узлах сетки.

Помогите фермеру определить максимально возможную площадь сарая.

**Входные данные**

В первой строке входного файла INPUT.TXT записаны два натуральных числа N и M ($1 \le N,M \le 1000$) – размеры фермы. Далее, следует N строк, в каждой из которых находится последовательность (без пробелов) из M нулей и единиц, описывающих ферму. Единицы соответствуют свободным для постройки участкам.

**Выходные данные**

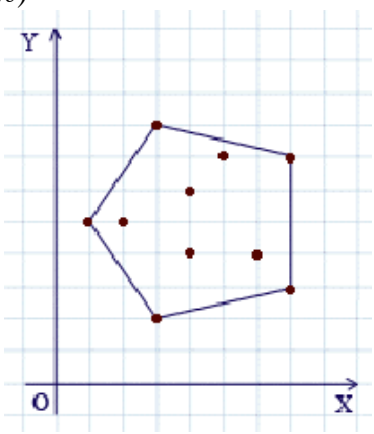
В выходной файл OUTPUT.TXT необходимо вывести максимально возможную площадь сарая, который может построить фермер на своем участке.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 5 10<br>1011011111<br>0111111110<br>1111111111<br>1011111111<br>1101110111 | 21 |

## 117. Опасная зона

*(Время: 1 сек. Память: 16 Мб Сложность: 57%)*




Группа экспертов обнаружила на территории нежилого массива множество опасных участков, соприкосновение с которыми небезопасно для жизни человека.

В целях безопасности требуется создать защитный периметр в форме выпуклого многоугольника, который бы смог обезопасить проникновение человека в эту зону.

По заданным координатам опасных участков требуется вычислить минимально возможную площадь опасной зоны, которая попадет в защитный периметр.

**Входные данные**

В первой строке входного файла INPUT.TXT записано натуральное число N – количество опасных участков. В каждой из N последующих строк находятся два числа $X_i$ и $Y_i$ – координаты участков, размерами которых можно пренебречь. При этом участки могут повторяться.

Все числа целые, не превосходящие 1000 по абсолютной величине.

**Выходные данные**

В выходной файл OUTPUT.TXT необходимо вывести одно число – площадь опасной зоны, округленной до целого значения.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 10<br>4 6<br>2 5<br>6 4<br>7 7<br>4 4<br>1 5<br>3 8<br>3 2<br>5 7<br>7 3 | 26 |

## 118. Задача Иосифа Флавия

*(Время: 1 сек. Память: 16 Мб Сложность: 19%)*

Существует легенда, что Иосиф Флавий – известный историк первого века – выжил и стал известным благодаря математической одаренности. В ходе иудейской войны он в составе отряда из 41 иудейского воина был загнан римлянами в пещеру. Предпочитая самоубийство плену, воины решили выстроиться в круг и последовательно убивать каждого третьего из живых до тех пор, пока не останется ни одного человека. Однако Иосиф наряду с одним из своих единомышленников счел подобный конец бессмысленным – он быстро вычислил спасительные места в порочном круге, на которые поставил себя и своего товарища. И лишь поэтому мы знаем его историю…

В нашем варианте мы начнем с того, что выстроим в круг N человек, пронумерованных числами от 1 до N, и будем исключать каждого k-ого до тех пор, пока не уцелеет только один человек.

Например, если N=10, K=3, то сначала умрет 3-ий, потом 6-ой, затем 9-ый, затем 2-ой, затем 7-ой, потом 1-ый, потом 8-ой, за ним – 5-ый, и потом 10-ый. Таким образом, уцелеет 4-ый.

Требуется написать программу, которая по заданным N и K будет определять номер уцелевшего человека.

**Входные данные**

Входной файл INPUT.TXT содержит два натуральных числа N и K. Ограничения: $N \le 500$, $K \le 100$.

**Выходные данные**

В выходной файл OUTPUT.TXT нужно вывести номер уцелевшего человека.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 10 3 | 4 |

## 119. Сортировка времени

*(Время: 1 сек. Память: 16 Мб Сложность: 13%)*

Требуется выполнить сортировку временных моментов, заданных в часах, минутах и секундах.

**Входные данные**

Во входном файле INPUT.TXT в первой строке записано число N ($1 \le N \le 100$), а в последующих N строках N моментов времени. Каждый момент времени задается 3 целыми числами – часы (от 0 до 23), минуты (от 0 до 59) и секунды (от 0 до 59).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите моменты времени, упорядоченные в порядке неубывания без ведущих нулей.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 4<br>10 20 30<br>7 30 00<br>23 59 59<br>13 30 30 | 7 30 0<br>10 20 30<br>13 30 30<br>23 59 59 |

**Разбор**

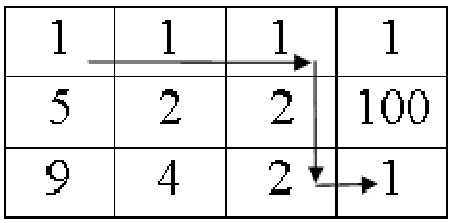
Эта задача сводится к сортировке массива. Удобнее всего эту задачу решать, переводя каждый момент времени в секунды. Так получим целочисленный массив, отсортировав который сможем вывести его, совершив обратное преобразование из секунд в часы, минуты и секунды. Для прямого и обратного перевода можно использовать следующие формулы:

```
x = 3600*h + 60*m + s
h = x div 3600
m = (x div 60) mod 60
s = x mod 60
```

Заметим так же, что здесь можно использовать любой алгоритм сортировки, т.к. элементов не более 100. В качестве типа для хранения элементов массива следует использовать 4-байтный целый тип, т.к. в сутках 86400 секунд.

## 120. Минимальный путь в таблице

*(Время: 1 сек. Память: 16 Мб Сложность: 32%)*




В прямоугольной таблице NxM (в каждой клетке которой записано некоторое число) в начале игрок находится в левой верхней клетке. За один ход ему разрешается перемещаться в соседнюю клетку либо вправо, либо вниз (влево и вверх перемещаться запрещено). При проходе через клетку с игрока берут столько у.е., какое число записано в этой клетке (деньги берут также за первую и последнюю клетки его пути).

Требуется найти минимальную сумму у.е., заплатив которую игрок может попасть в правый нижний угол.

**Входные данные**

Во входном файле INPUT.TXT задано два числа N и M – размеры таблицы ($1 \le N \le 20$, $1 \le M \le 20$). Затем идет N строк по M чисел в каждой – размеры штрафов в у.е. за прохождение через соответствующие клетки (числа от 0 до 100).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите минимальную сумму, потратив которую можно попасть в правый нижний угол.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3 4<br>1 1 1 1<br>5 2 2 100<br>9 4 2 1 | 8 |

## 121. Гвоздики

*(Время: 1 сек. Память: 16 Мб Сложность: 34%)*

На прямой дощечке вбиты гвоздики. Любые два гвоздика можно соединить ниточкой. Требуется соединить некоторые пары гвоздиков ниточками так, чтобы к каждому гвоздику была привязана хотя бы одна ниточка, а суммарная длина всех ниточек была минимальна.

**Входные данные**

В первой строке входного файла INPUT.TXT записано число N – количество гвоздиков ($1 \le N \le 100$). В следующей строке записано N чисел – координаты всех гвоздиков (неотрицательные целые числа, не превосходящие 10000).

**Выходные данные**

В выходной файл OUTPUT.TXT нужно вывести единственное число – минимальную суммарную длину всех ниточек.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 5<br>4 10 0 12 2 | 6 |

## 122. Максимальная подпоследовательность

*(Время: 1 сек. Память: 16 Мб Сложность: 38%)*

Дана числовая последовательность, требуется найти длину наибольшей возрастающей подпоследовательности.

**Входные данные**

В первой строке входного файла INPUT.TXT записано число N – длина последовательности ($1 \le N \le 1000$). Во второй строке записана сама последовательность (через пробел). Числа последовательности – целые числа, не превосходящие 10000 по модулю.

**Выходные данные**

В выходной файл OUTPUT.TXT требуется вывести наибольшую длину возрастающей подпоследовательности.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 6<br>3 29 5 5 28 6 | 3 |

## 123. Восстановление скобок

*(Время: 1 сек. Память: 16 Мб Сложность: 54%)*

Задан шаблон, состоящий из круглых скобок и знаков вопроса. Требуется определить, сколькими способами можно заменить знаки вопроса круглыми скобками так, чтобы получилось правильное скобочное выражение.

**Входные данные**

Единственная строка входного файла INPUT.TXT содержит заданный шаблон длиной не более 80 символов.

**Выходные данные**

Выведите в выходной файл OUTPUT.TXT искомое количество способов. Исходные данные будут таковы, что это количество не превзойдет $2*10^9$.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | ????(? | 2 |

## 124. Светофорчики

*(Время: 1 сек. Память: 16 Мб Сложность: 25%)*

В подземелье M тоннелей и N перекрестков, каждый тоннель соединяет какие-то два перекрестка. Мышиный король решил поставить по светофору в каждом тоннеле перед каждым перекрестком. Напишите программу, которая посчитает, сколько светофоров должно быть установлено на каждом из перекрестков. Перекрестки пронумерованы числами от 1 до N.

**Входные данные**

Во входном файле INPUT.TXT записано два числа N и M ($0 < N \le 100$, $0 \le M \le N*(N-1)/2$). В следующих M строках записаны по два числа i и j ($1 \le i, j \le N$), которые означают, что перекрестки i и j соединены тоннелем. Можно считать, что любые два перекрестка соединены не более чем одним тоннелем. Нет тоннелей от перекрестка i до него самого.

**Выходные данные**

В выходной файл OUTPUT.TXT вывести N чисел: k-ое число означает количество светофоров на k-ом перекрестке.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 7 10<br>5 1<br>3 2<br>7 1<br>5 2<br>7 4<br>6 5<br>6 4<br>7 5<br>2 1<br>5 3 | 3 3 2 2 5 2 3 |

## 125. Цветной дождь

*(Время: 1 сек. Память: 16 Мб Сложность: 26%)*

В Банановой республике очень много холмов, соединенных мостами. На химическом заводе произошла авария, в результате чего испарилось экспериментальное удобрение «зован». На следующий день выпал цветной дождь, причем он прошел только над холмами. В некоторых местах падали красные капли, в некоторых – синие, а в остальных – зеленые, в результате чего холмы стали соответствующего цвета. Президенту Банановой республики это понравилось, но ему захотелось покрасить мосты между вершинами холмов так, чтобы мосты были покрашены в цвет холмов, которые они соединяют. К сожалению, если холмы разного цвета, то покрасить мост таким образом не удастся. Посчитайте количество таких «плохих» мостов.

**Входные данные**

В файле INPUT.TXT в первой строке записано N ($0 < N \le 100$) – число холмов. Далее идет матрица смежности, описывающая наличие мостов между холмами (1 – мост есть, 0 – нет). Предпоследняя строка пустая, а в последней строке записано N чисел, обозначающих цвет холмов: 1 – красный; 2 – синий; 3 – зеленый.

**Выходные данные**

В файл OUTPUT.TXT вывести количество «плохих» мостов.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 7<br>0 1 0 0 0 1 1<br>1 0 1 0 0 0 0<br>0 1 0 0 1 1 0<br>0 0 0 0 0 0 0<br>0 0 1 0 0 1 0<br>1 0 1 0 1 0 0<br>1 0 0 0 0 0 0<br><br>1 1 1 1 1 3 3 | 4 |

## 126. Издевательство

*(Время: 1 сек. Память: 16 Мб Сложность: 28%)*

*Штирлиц ехал на машине, увидел голосующего Бормана, и проехал мимо. Через некоторое время он снова увидел голосующего Бормана, и снова проехал мимо. Вскоре он опять увидел голосующего Бормана. – Издевается! – подумал Борман. – Кольцевая! – догадался Штирлиц.*

В городе N площадей. Любые две площади соединены между собой ровно одной дорогой с двусторонним движением. В этом городе живет Штирлиц. У Штирлица есть хобби – он любит воскресным утром выйти из дома, сесть в машину, выбрать какой-нибудь кольцевой маршрут, проходящий ровно по трем площадям (то есть сначала он едет с какой-то площади на какую-то другую, потом – на третью, затем возвращается на начальную, и опять едет по этому маршруту). Он воображает, что где-то на этом пути стоит Борман. И так вот ездит Штирлиц все воскресенье, пока голова не закружится, и радуется...

Естественно, что Штирлицу хочется проезжать мимо точки, в которой, как он воображает, стоит Борман, как можно чаще. Для этого, естественно, выбранный Штирлицем маршрут должен быть как можно короче. Напишите программу, которая выберет оптимальный для Штирлица маршрут.

**Входные данные**

Во входном файле INPUT.TXT записано сначала число N ($3 \le N \le 100$), а затем матрица NxN расстояний между площадями (число в позиции i, j обозначает длину дороги, соединяющей i-ую и j-ую площади). Все числа в матрице (кроме стоящих на главной диагонали) – натуральные, не превышающие 1000. Матрица симметрична относительно главной диагонали, на главной диагонали стоят 0.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите длину оптимального маршрута.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 5<br>0 20 10 30 40<br>20 0 30 1 2<br>10 30 0 40 1000<br>30 1 40 0 21<br>40 2 1000 21 0 | 24 |

## 127. Путь

*(Время: 1 сек. Память: 16 Мб Сложность: 40%)*

В неориентированном графе требуется найти длину кратчайшего пути между двумя вершинами.

**Входные данные**

Во входном файле INPUT.TXT записано сначала число N – количество вершин в графе ($1 \le N \le 100$). Затем записана матрица смежности (0 обозначает отсутствие ребра, 1 – наличие ребра). Затем записаны номера двух вершин – начальной и конечной.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите длину кратчайшего пути. Если пути не существует, выведите одно число – 1.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 5<br>0 1 0 0 1<br>1 0 1 0 0<br>0 1 0 0 0<br>0 0 0 0 0<br>1 0 0 0 0<br>3 5 | 3 |

## 128. Один конь

*(Время: 1 сек. Память: 16 Мб Сложность: 43%)*

На шахматной доске NxN в клетке (x1, y1) стоит голодный шахматный конь. Он хочет попасть в клетку (x2, y2), где растет вкусная шахматная трава. Какое наименьшее количество ходов он должен для этого сделать?

**Входные данные**

Входной файл INPUT.TXT содержит пять чисел: N, x1, y1, x2, y2 ($5 \le N \le 20$, $1 \le x1, y1, x2, y2 \le N$). Левая верхняя клетка доски имеет координаты (1, 1), правая нижняя - (N, N).

**Выходные данные**

В выходной файл OUTPUT.TXT необходимо вывести наименьшее число ходов коня.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 5<br>1 1<br>3 1 | 2 |

## 129. Табличка

*(Время: 1 сек. Память: 16 Мб Сложность: 40%)*

Вам дана табличка, состоящая из N строк и M столбцов. В каждой клетке таблицы стоит либо 0, либо 1. Расстоянием между клетками (x1, y1) и (x2, y2) называется |x1-x2|+|y1-y2|. Вам нужно построить другую таблицу, в которой в каждой клетке стоит расстояние от данной до ближайшей клетки, содержащей 1 (в начальной таблице). Гарантируется, что хотя бы одна 1 в таблице есть.

**Входные данные**

В первой строке входного файла INPUT.TXT содержатся два натуральных числа, не превосходящих 100 - N и M. Далее идут N строк по M чисел – элементы таблицы.

**Выходные данные**

Выходной файл OUTPUT.TXT должен содержать N строк по M чисел – элементы искомой таблицы.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 2 3<br>0 0 1<br>1 0 0 | 1 1 0<br>0 1 1 |

## 130. Два коня

*(Время: 1 сек. Память: 16 Мб Сложность: 55%)*

На стандартной шахматной доске (8x8) живут 2 шахматных коня: красный и зеленый. Обычно они беззаботно скачут по просторам доски, пощипывая шахматную травку, но сегодня особенный день: у зеленого коня день рождения. Зеленый конь решил отпраздновать это событие вместе с красным. Но для осуществления этого прекрасного плана им нужно оказаться на одной клетке. Заметим, что красный и зеленый шахматные кони сильно отличаются от черного с белым: они ходят не по очереди, а одновременно, и если оказываются на одной клетке, никто никого не съедает. Сколько ходов им потребуется, чтобы насладиться праздником?

**Входные данные**

Во входном файле INPUT.TXT содержатся координаты коней, записанные по стандартным шахматным правилам (т.е. двумя символами – маленькая латинская буква (от a до h) и цифра (от 1 до 8), задающие столбец и строку соответственно).

**Выходные данные**

Выходной файл OUTPUT.TXT должен содержать наименьшее необходимое количество ходов, либо – 1, если кони не могут встретиться.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | a1 a3 | 1 |

## 131. Перепись

*(Время: 1 сек. Память: 16 Мб Сложность: 15%)*

В доме живет N жильцов. Однажды решили провести перепись всех жильцов данного дома и составили список, в котором указали возраст и пол каждого жильца. Требуется найти номер самого старшего жителя мужского пола.

**Входные данные**

Во входном файле INPUT.TXT в первой строке задано натуральное число N – количество жильцов (N≤100). В последующих N строках располагается информация о всех жильцах: каждая строка содержит два целых числа: V и S – возраст и пол человека ($1 \le V \le 100$, S - 0 или 1). Мужскому полу соответствует значение S=1, а женскому – S=0.

**Выходные данные**

Выходной файл OUTPUT.TXT должен содержать номер самого старшего мужчины в списке. Если таких жильцов несколько, то следует вывести наименьший номер. Если жильцов мужского пола нет, то выведите -1.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 4<br>25 1<br>70 1<br>100 0<br>3 1 | 2 |

## 132. Алгоритм Дейкстры

*(Время: 1 сек. Память: 16 Мб Сложность: 47%)*

Дан ориентированный взвешенный граф. Для него вам необходимо найти кратчайшее расстояние от вершины S до вершины F.

**Входные данные**

В первой строке входного файла INPUT.TXT записаны три числа: N, S и F ($1 \le N \le 100$; $1 \le S, F \le N$), где N – количество вершин графа. В следующих N строках записаны по N чисел – матрица смежности графа, где число в i-ой строке j-ом столбце соответствует ребру из i в j: -1 означает отсутствие ребра между вершинами, а любое неотрицательное число – наличие ребра данного веса. На главной диагонали матрицы всегда записаны нули.

**Выходные данные**

В выходной файл OUTPUT.TXT необходимо вывести искомое расстояние или -1, если пути между указанными вершинами не существует.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3 1 2<br>0 -1 2<br>3 0 -1<br>-1 4 0 | 6 |

## 133. Заправки

*(Время: 1 сек. Память: 16 Мб Сложность: 49%)*

В стране N городов, некоторые из которых соединены между собой дорогами. Для того, чтобы проехать по одной дороге требуется один бак бензина. В каждом городе бак бензина имеет разную стоимость. Вам требуется добраться из первого города в N-ый, потратив как можно меньшее количество денег.

**Входные данные**

Во входном файле INPUT.TXT записано сначала число N ($1 \le N \le 100$), затем идет N чисел, i-ое из которых задает стоимость бензина в i-ом городе (все числа целые из диапазона от 0 до 100). Далее идет число M – количество дорог в стране, далее идет описание самих до-

рог. Каждая дорога задается двумя числами – номерами городов, которые она соединяет. Все дороги двухсторонние (то есть по ним можно ездить как в одну, так и в другую сторону); между двумя городами всегда существует не более одной дороги; не существует дорог, ведущих из города в себя.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите одно число – суммарную стоимость маршрута или -1, если добраться невозможно.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 4<br>1 10 2 15<br>4<br>1 2<br>1 3<br>4 2<br>4 3 | 3 |

**Пояснение к примеру**

Оптимальное решение в примере: из 1-го города поехать в 3-ий, а затем в 4-ый. Горючее придется покупать в 1 и 3 городах.

## 134. Автобусы

*(Время: 1 сек. Память: 16 Мб Сложность: 50%)*

Между некоторыми деревнями края Власюки ходят автобусы. Поскольку пассажиропотоки здесь не очень большие, то автобусы ходят всего несколько раз в день.

Марии Ивановне требуется добраться из деревни d в деревню v как можно быстрее (считается, что в момент времени 0 она находится в деревне d).

**Входные данные**

Во входном файле INPUT.TXT записано число N – общее число деревень ($1 \le N \le 100$), номера деревень d и v, затем количество автобусных рейсов R ($0 \le R \le 10000$). Затем идут описания автобусных рейсов. Каждый рейс задается номером деревни отправления, временем отправления, деревней назначения и временем прибытия (все времена – целые от 0 до 10000). Если в момент t пассажир приезжает в деревню, то уехать из нее он может в любой момент времени, начиная с t.

**Выходные данные**

В выходной файл OUTPUT.TXT вывести минимальное время, когда Мария Ивановна может оказаться в деревне v. Если он не сможет с помощью указанных автобусных рейсов добраться из d в v, вывести -1.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3<br>1 3<br>4<br>1 0 2 5<br>1 1 2 3<br>2 3 3 5<br>1 1 3 10 | 5 |

## 135. Алгоритм Флойда

*(Время: 1 сек. Память: 16 Мб Сложность: 36%)*

Полный ориентированный взвешенный граф задан матрицей смежности. Постройте матрицу кратчайших путей между его вершинами. Гарантируется, что в графе нет циклов отрицательного веса.

**Входные данные**

В первой строке входного файла INPUT.TXT записано единственное число N ($1 \le N \le 100$) – количество вершин графа. В следующих N строках по N чисел – матрица смежности графа (j-ое число в i-ой строке соответствует весу ребра из вершины i в вершину j). Все числа по модулю не превышают 100. На главной диагонали матрицы – всегда нули.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите N строк по N чисел – матрицу кратчайших расстояний между парами вершин. j-ое число в i-ой строке должно быть равно весу кратчайшего пути из вершины i в вершину j.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 4<br>0 5 9 100<br>100 0 2 8<br>100 100 0 7<br>4 100 100 0 | 0 5 7 13<br>12 0 2 8<br>11 16 0 7<br>4 9 11 0 |

## 136. Алгоритм Флойда - 2

*(Время: 1 сек. Память: 16 Мб Сложность: 39%)*

Дан ориентированный взвешенный граф. Вам необходимо найти пару вершин, кратчайшее расстояние от одной из которых до другой максимально среди всех пар вершин.

**Входные данные**

В первой строке входного файла INPUT.TXT записано единственное число N ($1 \le N \le 100$) – количество вершин графа. В следующих N строках по N чисел – матрица смежности графа, где -1 означает отсутствие ребра между вершинами, а любое неотрицательное число – присутствие ребра данного веса. На главной диагонали матрицы – всегда нули.

**Выходные данные**

В выходной файл OUTPUT.TXT требуется вывести искомое максимальное кратчайшее расстояние.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 4<br>0 5 9 -1<br>-1 0 2 8<br>-1 -1 0 7<br>4 -1 -1 0 | 16 |

## 137. Существование пути

*(Время: 1 сек. Память: 16 Мб Сложность: 65%)*

Дан ориентированный взвешенный граф. По его матрице смежности нужно для каждой пары вершин определить: существует кратчайший путь между ними или нет.

Кратчайший путь может не существовать по двум причинам: либо нет ни одного пути, либо есть путь сколь угодно маленького веса.

**Входные данные**

В первой строке входного файла INPUT.TXT записано единственное число N ($1 \le N \le 100$) – количество вершин графа. В следующих N строках по N чисел – матрица смежности графа (j-ое число в i-ой строке соответствует весу ребра из вершины i в вершину j), в которой число 0 обозначает отсутствие ребра, а любое другое число – наличие ребра соответствующего веса. Все числа по модулю не превышают 100.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите N строк по N чисел: j-ое число в i-ой строке должно быть равно 0, если путь из i в j не существует, 1 – если существует кратчайший путь, и 2 – если существует путь сколь угодно маленького веса.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 5<br>0 1 2 0 0<br>1 0 3 0 0<br>2 3 0 0 0<br>0 0 0 0 -1<br>0 0 0 -1 0 | 1 1 1 0 0<br>1 1 1 0 0<br>1 1 1 0 0<br>0 0 0 2 2<br>0 0 0 2 2 |

## 138. Алгоритм Форда-Беллмана

*(Время: 1 сек. Память: 16 Мб Сложность: 38%)*

Дан ориентированный граф, в котором могут быть кратные ребра и петли. Каждое ребро имеет вес, выражающийся целым числом (возможно, отрицательным). Гарантируется, что циклы отрицательного веса отсутствуют.

Требуется посчитать длины кратчайших путей от вершины номер 1 до всех остальных вершин.

**Входные данные**

Во входном файле INPUT.TXT записано сначала число N ($1 \le N \le 100$) – количество вершин графа, далее идет число M ($0 \le M \le 10000$) – количество ребер. Далее идет M троек чисел, описывающих ребра: начало ребра, конец ребра и вес (вес – целое число от -100 до 100).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите N чисел – расстояния от вершины номер 1 до всех вершин графа. Если пути до соответствующей вершины не существует, вместо длины пути выведите число 30000.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 4 5<br>1 2 10<br>2 3 10<br>1 3 100<br>3 1 -10<br>2 3 1 | 0 10 11 30000 |

## 139. Лабиринт знаний

*(Время: 1 сек. Память: 16 Мб Сложность: 45%)*

В стране Умландии построили аттракцион «Лабиринт знаний». Лабиринт представляет собой N комнат, занумерованных от 1 до N, между некоторыми из которых есть двери. Когда человек проходит через дверь, показатель его знаний изменяется на определенную величину, фиксированную для данной двери. Вход в лабиринт находится в комнате 1, выход – в комнате N. Каждый ученик проходит лабиринт ровно один раз и попадает в ту или иную учебную группу в зависимости от количества набранных знаний (при входе в лабиринт этот показатель равен нулю). Ваша задача показать наилучший результат.

**Входные данные**

Первая строка входного файла INPUT.TXT содержит целые числа N ($1 \le N \le 2000$) – количество комнат и M ($1 \le M \le 10000$) – количество дверей. В каждой из следующих M строк содержится описание двери – номера комнат, из которой она ведет и в которую она ведет (через дверь можно ходить только в одном направлении), а также целое число, которое

прибавляется к количеству знаний при прохождении через дверь (это число по модулю не превышает 10000). Двери могут вести из комнаты в нее саму, между двумя комнатами может быть более одной двери.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите: ":)" – если можно получить неограниченно большой запас знаний, ":(" – если лабиринт пройти нельзя, и максимальное количество набранных знаний в противном случае.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 2 2<br>1 2 5<br>1 2 -5 | 5 |

## 140. Цикл отрицательного веса

*(Время: 1 сек. Память: 16 Мб Сложность: 46%)*

Дан взвешенный граф. Определить, есть ли в нем цикл отрицательного веса.

**Входные данные**

Во входном файле INPUT.TXT в первой строке записано число N ($1 \le N \le 100$) – количество вершин графа. В следующих N строках находится по N чисел – матрица смежности графа. Веса ребер не превышают по модулю 10000. Если ребра нет, соответствующее значение равно 100000.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите «YES», если цикл существует, или «NO» в противном случае.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 2<br>0 -1<br>-1 0 | YES |

## 141. Дерево

*(Время: 1 сек. Память: 16 Мб Сложность: 42%)*

Неориентированный граф без петель и кратных ребер задан матрицей смежности. Требуется определить, является ли этот граф деревом.

**Входные данные**

Во входном файле INPUT.TXT записано сначала число N – количество вершин графа (от 1 до 100). Далее записана матрица смежности размером N*N, в которой 1 обозначает наличие ребра, 0 – его отсутствие. Матрица симметрична относительно главной диагонали.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите сообщение YES, если граф является деревом, и NO в противном случае.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3<br>0 1 0<br>1 0 1<br>0 1 0 | YES |

## 142. Минимальный каркас

*(Время: 1 сек. Память: 16 Мб Сложность: 53%)*

От вас требуется определить вес минимального остовного дерева для неориентированного взвешенного связного графа.

**Входные данные**

В первой строке входного файла INPUT.TXT находятся числа N и M ($1 \le N \le 100$; $1 \le M \le 6000$), где N – количество вершин в графе, а M – количество рёбер. В каждой из последующих M строк записано по тройке чисел A, B, C, где A и B – номера вершин, соединённых ребром, а C – вес ребра (натуральное число, не превышающее 30000).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите одно число – искомый вес.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3 3<br>1 2 1<br>2 3 2<br>3 1 3 | 3 |

## 143. A-B

*(Время: 1 сек. Память: 16 Мб Сложность: 44%)*

Требуется найти разность между неотрицательными числами A и B.

**Входные данные**

Во входном файле INPUT.TXT в двух строках записаны два неотрицательных целых числа A и B, не превышающие $10^{1000}$.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите значение A-B.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 7<br>5 | 2 |
| 2 | 5<br>17 | -12 |

**Разбор**

При вычетании длинных чисел удобно вычитать из наибольшего наименьшее, поэтому предварительно нужно их сравнить и в случае необходимости поменять местами. Так как A-B=-(B-A), то в случае когда A < B можно вывести знак "-", а затем положительное значение (B-A). При вычитании будем выполнять поразрядное вычитание с использованием переменной c для хранения значения переноса. Результат операции можно сохранять в массиве a, т.к. ранее используемые разряды фактически не имеют значения для дальнейшего процесса вычисления.

Опишем решение данной задачи в виде следующего алгоритма:

```
const maxsize=1001;
int a[maxsize], b[maxsize];

readlong(a);
readlong(b);

if(complong(a,b) < 0){
  write('-');
  a <--> b;
}

c=0;
for i=1..a[0]{
  c = c+a[i]-b[i]+10;
  a[i] = c mod 10;
  if(c < 10) c=-1; else c=0;
}
while(a[a[0]]=0 and a[0]>1) a[0]=a[0]-1;

writelong(a);
```

## 144. A*B

*(Время: 1 сек. Память: 16 Мб Сложность: 37%)*

Даны два целых неотрицательных числа A и B. Требуется найти их произведение.

**Входные данные**

Во входном файле INPUT.TXT записаны целые неотрицательные числа A и B по одному в строке ($A < 10^{100}$, $B \le 10000$).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите единственное число без лидирующих нулей - A*B.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 5<br>7 | 35 |

**Разбор**

В данной задаче требуется умножить длинное число на короткое. Когда мы делаем подобное вручную, то мы умножаем на короткое число поразрядно (т.е. поциферно), но при программной реализации мы можем число A умножать на B так же, как если бы B состояло всего из одной цифры. В процессе умножения результат будем записывать сразу в имеющийся массив a, а значение переноса в переменную c.

Приведем один из алгоритмов решения данной задачи:

```
const maxsize=105;
int a[maxsize], b, c=0;

readlong(a);
read(b);

for i=1..a[0]{
  a[i] = a[i]*b+c;
  c = a[i] div 10;
  a[i] = a[i] mod 10;
}
while(c>0){
  a[0] = a[0]+1;
  a[a[0]] = c mod 10;
  c = c div 10;
}

writelong(a);
```

## 145. A div B

*(Время: 1 сек. Память: 16 Мб Сложность: 40%)*

Даны два натуральных числа A и B. Требуется найти их целую часть от их частного.

**Входные данные**

Во входном файле INPUT.TXT записаны натуральные числа A и B по одному в строке ($A < 10^{100}$, $B \le 10000$).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите единственное число без лидирующих нулей - A div B.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 7 3 | 2 |

**Разбор**

Рассмотрим процесс деления длинного числа на короткое. Будем руководствоваться теми же принципами, что и при делении «в столбик». Будем в некоторой переменной хранить текущее значение, по которому будет определяться последующая цифра результата, ко-

торая получается из целочисленного деления текущего значения на делитель. Далее вычисляем следующее текущее значение как (x mod b)*10+r, где x-старое текущее значение, b – последняя вычисленная цифра, а r – следующая цифра делимого. Процесс продолжается до последней цифры длинного числа.

Алгоритм, реализующий данную задачу, может быть записан в следующем виде:

```
const maxsize=101;
int a[maxsize], b;

readlong(a);
read(b);

x=0; k=0;
for i=a[0]..1{
  x = x*10+a[i];
  if(x < b and k=0 and i > 1) continue;
  k=1;
  write(x div b);
  x = x mod b;
}
```

## 146. Длинный корень

*(Время: 1 сек. Память: 16 Мб Сложность: 67%)*

По заданному натуральному числу A требуется найти наибольшее число B такое, что $B^2 \le A$.

**Входные данные**

Во входном файле INPUT.TXT записано натуральное число A ($A \le 10^{3000}$).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите максимальное натуральное число B, квадрат которого не превосходит A. Число B следует выводить без лидирующих нулей.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 17 | 4 |

**Разбор**

Для начала найдем количество цифр в ответе. Покажем, что квадрат K-значного числа имеет либо 2*K-1, либо 2*K цифр. Действительно, квадрат наименьшего K-значного числа, т.е. числа $10^{K-1}$, равен $10^{2K-2}$, т.е. имеет 2*K-1 цифру; квадрат наибольшего K-значного числа, т.е. числа $10^K$-1, равен $10^{2K}$-2*$10^K$+1, т.е. имеет 2*K цифр. Поскольку функция $y=x^2$ монотонно возрастает для всех x>0(т.е. для любых $x_1,x_2 > 0$, таких что $x_1 < x_2$ выполнено $x_1^2 < x_2^2$), квадрат любого K-значного числа будет иметь либо 2*K-1, либо 2*K цифр. Следовательно, если число A из входного файла имеет N цифр, то корень из него будет иметь ровно (N+1) div 2 цифр.

Теперь, когда мы знаем количество цифр в числе, будем последовательно подбирать его цифры, начиная со старших. Пусть K старших цифр уже подобраны. Поставим на K+1 место самую большую цифру - 9, и будем уменьшать ее до тех пор, пока квадрат полученного таким образом числа (считая, что все цифры ответа, начиная с K+2 и до самой младшей равны 0) не станет меньше либо равен числу A из входного файла. Таким образом, мы подобрали K+1 цифру нашего числа. Продолжая этот процесс, получим ответ на поставленную задачу.

В изложенном решении используется операция умножения «длинных» чисел. Благодаря алгоритму умножения «в столбик» эта задача сводится к многократному умножению «длинных» чисел на «короткие» и сложению «длинных» чисел. Заметим, что эта операция легко выполняется в том случае, если одно из чисел является степенью 10, умноженной на «короткое» число. Тогда достаточно умножить «длинное» число на это короткое, а затем сдвинуть результат на нужное число позиций, что равносильно умножению числа на степень 10.

Пусть aK – число, полученное на K-м шаге нашего приближения, b – очередная цифра, умноженная на 10 в соответствующей степени. Тогда $a_{K+1}^2=(a_K+b)^2=a_K^2+2a_Kb+b^2$. В стоящей справа сумме все слагаемые, кроме первого, представляют собой частный случай перемножения «длинных» чисел, изложенный выше. А первое слагаемое уже было вычисленно на предыдущем шаге алгоритма.

*Разбор задачи взят с сайта http://olympiads.ru.*

## 147. Числа Фибоначчи

*(Время: 1 сек. Память: 16 Мб Сложность: 16%)*

Последовательностью Фибоначчи называется последовательность чисел $a_0$, $a_1$, ..., $a_n$, ..., где $a_0 = 0$, $a_1 = 1$, $a_k = a_{k-1} + a_{k-2}$ ($k > 1$).

Требуется найти N-е число Фибоначчи.

**Входные данные**

Во входном файле INPUT.TXT записано целое число N ($0 \le N \le 30$).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите N-е число Фибоначчи.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 7 | 13 |

**Разбор**

Числа Фибоначчи имеют ряд замечательных свойств. Далее рассмотрим одно из решений этой задачи:

```
int a=0,b=1,c;
read(n);
for i=2..n{
  c=a+b;
  a=b;
  b=c;
}
if(i<2) write(i) else write(c);
```

Предствим рекурсивную реализацию данной задачи, которая работает значительно медленнее (подумайте почему), но уместна в ограничениях данной задачи:

```
int fib(int n){
  if(n<2) return n
     else return fib(n-1)+fib(n-2);
}
int n;
read(n);
write(fib(n));
```

## 148. НОД

*(Время: 1 сек. Память: 16 Мб Сложность: 15%)*

Даны два натуральных числа A и B. Требуется найти их наибольший общий делитель (НОД).

**Входные данные**

Во входном файле INPUT.TXT в единственной строке записаны натуральные числа A и B через пробел ($A, B \le 10^9$).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите НОД чисел A и B.

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 12 42 | 6 |

**Разбор**

Напомним, что *наибольшим общим делителем (НОД)* для целых чисел a и b называют такое наибольшее число d, что a делится на d и b делится на d. При этом записывают это так: (a,b)=1 или НОД(a,b)=d или GCD(a,b)=d. Числа a и b называют *взаимнопростыми*, когда НОД(a,b)=1. Данное определение обобщается и на набор чисел. При этом заметим, что попарной простоты недостаточно для подтверждения взаимной простоты набора чисел.

Наиболее функциональным и практичным методом поиска НОД является *алгортим Евклида*, который заключается в следующем: используя свойство НОД(a,b)=НОД(a, a mod b) мы можем свести данную запись к виду НОД(d,0), где d – искомый наибольший делитель. Другими словами, пока у нас оба числа a и b больше нуля следует наибольшее из них заменять на остаток от деления наибольшего на наименьшее, достаточно быстро мы получим таким образом решение задачи.

Реализация алгоритма Евклида для поиска НОД на алгоритмическом языке может выглядеть следующим образом:

```
read(a,b);
while a*b > 0 do
  if a >= b then a = a mod b else b = b mod a;
write(a+b);
```

Наиболее красивая запись (но вовсе не самая понятная) решения этой задачи выглядит на языке C++ так:

```
#include <stdio.h>
#include <iostream.h>
int a,b;
int main(){
  freopen("input.txt","r",stdin);
  freopen("output.txt","w",stdout);
  cin>>a>>b;
  while(b) b^=a^=b^=a%=b;
  cout<<a;
  return 0;
}
```

## 149. Разворот

*(Время: 1 сек. Память: 16 Мб Сложность: 9%)*

Дано натуральное число N и последовательность из N элементов. Требуется вывести эту последовательность в обратном порядке.

**Входные данные**

В первой строке входного файла INPUT.TXT записано натуральное число N ($N \le 10^3$). Во второй строке через пробел идут N целых чисел, по модулю не превосходящих $10^3$ – элементы последовательности.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите заданную последовательность в обратном порядке.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3<br>1 2 3 | 3 2 1 |

**Разбор**

Эта простая задача сводится к чтению чисел в массив и выводу массива в обратном порядке. Обратный порядок существенно повышает желание использовать массив, т.к. возникает необходимость в запоминании всех элементов (если бы порядок вывода был прямым, то можно было бы считывая элемент, тут же его выводить).

Приведем нехитрый алгоритм решения этой задачи:

```
//читаем количество элементов в n
read(n);
//чтение элементов из файла в массив
for i=1..n{
  read(a[i]);
}
//вывод элементов из массива в файл в обратном порядке
for i=n..1{
  write(a[i],' ');
}
```

На самом деле эту задачу можно решить и без массива, используя рекурсию, где все элементы можно хранить в стеке рекурсии, но это значительно сложнее и требует понимания рекурсии. Если вам знаком этот механизм, то более полезно решить данную задачу именно таким образом.

## 150. Друзья

*(Время: 1 сек. Память: 16 Мб Сложность: 41%)*

В клубе N человек. Многие из них – друзья. Так же известно, что друзья друзей так же являются друзьями. Требуется выяснить, сколько всего друзей у конкретного человека в клубе.

**Входные данные**

В первой строке входного файла INPUT.TXT заданы два числа: N и S ($1 \le N \le 100$; $1 \le S \le N$), где N – количество человек в клубе, а S – номер конкретного человека. В следующих N строках записано по N чисел – матрица смежности, состоящая из единиц и нулей. Причем единица, стоящая в i-й строке и j-м столбце гарантирует, что люди с номерами i и j – друзья, а 0 – выражает неопределенность.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите количество гарантированных друзей у человека с номером S, помня о транзитивности дружбы.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3 1<br>0 1 0<br>1 0 1<br>0 1 0 | 2 |

## 151. Банкет

*(Время: 1 сек. Память: 16 Мб Сложность: 54%)*

На банкет были приглашены N Очень Важных Персон (ОВП). Были поставлены 2 стола. Столы достаточно большие, чтобы все посетители банкета могли сесть за любой из них. Проблема заключается в том, что некоторые ОВП не ладят друг с другом и не могут сидеть за одним столом. Вас попросили определить, возможно ли всех ОВП рассадить за двумя столами.

**Входные данные**

В первой строке входного файла INPUT.TXT дано два числа: N и M ($1 \le N, M \le 100$), где N – количество ОВП, а M – количество пар ОВП, которые не могут сидеть за одним столом. В следующих M строках записано по 2 числа – пары ОВП, которые не могут сидеть за одним столом.

**Выходные данные**

Если способ рассадить ОВП существует, то в выходной файл OUTPUT.TXT выведите YES и NO в противном случае.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3 2<br>1 2<br>1 3 | YES |

## 152. Построение

*(Время: 1 сек. Память: 16 Мб Сложность: 62%)*

Группа солдат-новобранцев прибыла в армейскую часть N666. После знакомства с прапорщиком стало очевидно, что от работ на кухне по очистке картофеля спасти солдат может только чудо.

Прапорщик, будучи не в состоянии запомнить фамилии, пронумеровал новобранцев от 1 до N. После этого он велел им построиться по росту (начиная с самого высокого). С этой несложной задачей могут справиться даже совсем необученные новобранцы, да вот беда, прапорщик уверил себя, что знает про некоторых солдат, кто из них кого выше, и это далеко не всегда соответствует истине.

После трех дней обучения новобранцам удалось выяснить, что знает (а точнее, думает, что знает) прапорщик. Помогите им, используя эти знания, построиться так, чтобы товарищ прапорщик остался доволен.

**Входные данные**

Во входном файле INPUT.TXT сначала идут числа N и M ($1 \le N \le 100$, $1 \le M \le 5000$) - количество солдат в роте и количество пар солдат, про которых прапорщик знает, кто из них выше. Далее идут эти пары чисел A и B по одной на строке ($1 \le A, B \le N$), что означает, что, по мнению прапорщика, солдат A выше, чем B.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите «Yes» если можно построиться так, чтобы прапорщик остался доволен и «No» если нельзя.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 5 4<br>1 3<br>1 4<br>4 3<br>5 2 | Yes |

## 153. Монетки - 2

*(Время: 1 сек. Память: 16 Мб Сложность: 51%)*

В волшебной стране используются монетки достоинством A1, A2,..., AM. волшебный человечек пришел в магазин и обнаружил, что у него есть ровно по две монетки каждого достоинства. Ему нужно заплатить сумму N. Напишите программу, определяющую, сможет ли он расплатиться без сдачи.

**Входные данные**

Во входном файле INPUT.TXT записано сначала число N ($1 \le N \le 10^9$), затем – число M ($1 \le M \le 15$) и далее M попарно различных чисел A1, A2, ..., AM ($1 \le Ai \le 10^9$).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите количество монет, которое придется отдать волшебному человечку, если он сможет заплатить указанную сумму без сдачи. Если решений несколько, выведите вариант, в котором волшебный человек отдаст наименьшее возможное количество монет. Если без сдачи не обойтись, то выведите одно число 0. Если же у волшебного человечка не хватит денег, чтобы заплатить указанную сумму, выведите одно число -1 (минус один).

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 5 2<br>1 2 | 3 |
| 2 | 7 2<br>1 2 | -1 |
| 3 | 5 2<br>3 4 | 0 |

## 154. Сумма кубов

*(Время: 1 сек. Память: 16 Мб Сложность: 52%)*

Известно, что любое натуральное число можно представить в виде суммы не более чем четырех квадратов натуральных чисел. Вася решил придумать аналогичное утверждение для кубов – он хочет узнать, сколько кубов достаточно для представления любого числа. Его первая рабочая гипотеза – восемь.

Выяснилось, что почти все числа, которые Вася смог придумать, представляются в виде суммы не более чем восьми кубов. Однако число 239, например, не допускает такого представления. Теперь Вася хочет найти какие-либо другие такие числа, а также, возможно, какую-либо закономерность в представлениях всех остальных чисел, чтобы выдвинуть гипотезу относительно вида всех чисел, которые не представляются в виде суммы восьми кубов.

Помогите Васе написать программу, которая проверяла бы, возможно ли представить данное натуральное число в виде суммы не более чем восьми кубов натуральных чисел, и если это возможно, то находила бы какое-либо такое представление.

**Входные данные**

Во входном файле INPUT.TXT записано натуральное число N ($1 \le N \le 2*10^9$).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите не более восьми натуральных чисел в порядке невозрастания, кубы которых в сумме дают N. Если вариантов несколько, то следует вывести тот, в котором сумма этих чисел минимальна. Если искомого представления не существует, то в выходной файл необходимо вывести слово IMPOSSIBLE.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 17 | 2 2 1 |
| 2 | 239 | IMPOSSIBLE |

## 155. Резисторы

*(Время: 1 сек. Память: 16 Мб Сложность: 59%)*

Радиолюбитель Петя решил собрать детекторный приемник. Для этого ему понадобился конденсатор емкостью C мкФ. В распоряжении Пети есть набор из n конденсаторов, емкости которых равны C1, C2, ... ,Cn соответственно. Петя помнит, как вычисляется емкость параллельного соединений двух конденсаторов (Cnew = C1 + C2) и последовательного соединения двух конденсаторов (Cnew = (C1*C2)/(C1+C2) ). Петя хочет спаять некоторую последовательно-параллельную схему из имеющегося набора конденсаторов, такую, что ее емкость ближе всего к искомой (то есть абсолютная величина разности значений минимальна). Разумеется, Петя не обязан использовать для изготовления схемы все конденсаторы.

Напомним определение последовательно-параллельной схемы. Схема, составленная из одного конденсатора, – последовательно-параллельная схема. Любая схема, полученная последовательным соединением двух последовательно-параллельных схем, – последовательно-параллельная, а также любая схема, полученная параллельным соединением двух последовательно-параллельных схем, – последовательно-параллельная.

**Входные данные**

В первой строке каждого входного файла INPUT.TXT заданы числа n и C. Во второй строке содержится последовательность емкостей имеющихся в наличии конденсаторов C1, C2, ..., Cn. Значения всех емкостей – вещественные числа. Для всех входных файлов n < 7.

**Выходные данные**

В выходной файл OUTPUT.TXT необходимо вывести «YES», если Пете удастся собрать схему, емкость которой отличается не более чем на 0.01 от требуемого значения C.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3 1.66<br>1 2 1 | YES |

**Пояснения к примеру**

Последовательно соединим первый и второй конденсаторы, а затем полученную схему соединим параллельно с третьим. Полученная схема будет иметь емкость 1.(6)

## 156. Шахматы - 2

*(Время: 1 сек. Память: 16 Мб Сложность: 46%)*

Требуется найти число способов расставить на шахматной доске NxN K ладей так, чтобы они не били друг друга. Все ладьи считаются одинаковыми.

**Входные данные**

Во входном файле INPUT.TXT записаны натуральные числа N и K (N, K ≤ 8).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите одно целое число – ответ на задачу.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 8 8 | 40320 |

## 157. Карточки

*(Время: 1 сек. Память: 16 Мб Сложность: 48%)*

На день рождения Пете подарили набор карточек с буквами. Теперь Петя с большим интересом составляет из них разные слова. И вот, однажды, составив очередное слово, Петя заинтересовался вопросом: «А сколько различных слов можно составить из тех же карточек, что и данное?».

Помогите ему ответить на этот вопрос.

**Входные данные**

Во входном файле INPUT.TXT задано слово, составленное Петей – строка из маленьких латинских букв не длиннее 15 символов.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите одно целое число – ответ на поставленную задачу.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | solo | 12 |

## 158. Великий комбинатор

*(Время: 1 сек. Память: 16 Мб Сложность: 60%)*

В результате очередной хитроумной комбинации у Остапа Бендера и его компаньонов - K детей лейтенанта Шмидта оказалось X рублей пятирублевыми банкнотами. И вот дело, как водится, дошло до дележа...

Шура Балаганов предложил «по справедливости», т.е. всем поровну. Паниковский порешил себе отдать половину, а остальным «по заслугам». Каждый из K детей лейтенанта предложил что-нибудь интересное. Однако у Великого Комбинатора имелось свое мнение на этот счет...

Ваша же задача состоит в нахождении количества способов разделить имеющиеся деньги между всеми участниками этих славных событий: K детьми лейтенанта Шмидта и Остапом Бендером.

**Входные данные**

Во входном файле INPUT.TXT записаны целые числа X ($0 \le X \le 500$) и K ($0 \le K \le 100$). Естественно, что число X делится на 5. Да и при дележе рвать пятирублевые банкноты не разрешается.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите одно целое число - количество способов дележа.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 15 2 | 10 |

## 159. Обратная перестановка

*(Время: 1 сек. Память: 16 Мб Сложность: 25%)*

По заданной перестановке требуется определить обратную.

*Перестановкой* из N элементов называется упорядоченный набор из N различных чисел от 1 до N. Количество различных перестановок порядка N равно $P_N = N!$

Пусть у нас есть упорядоченное множество из N элементов. Перестановка задает преобразование этого множества. А именно, она говорит, что на i место нужно поставить $a_i$ элемент множества, где $a_i$ - i-тый элемент перестановки.

*Обратной перестановкой* к перестановке $\pi$ называется такая перестановка $\pi^{-1}$, что $\pi\pi^{-1} = \pi^{-1}\pi = \varepsilon$, где $\varepsilon$ – тождественная перестановка.

**Входные данные**

В первой строке входного файла INPUT.TXT записано число $0 < N \le 20000$ - порядок перестановки. Во второй строке записана сама перестановка.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите обратную перестановку.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3<br>2 3 1 | 3 1 2 |

## 160. Степень перестановки

*(Время: 1 сек. Память: 16 Мб Сложность: 46%)*

Требуется вычислить степень заданной перестановки.

*Перестановкой* из N элементов называется упорядоченный набор из N различных чисел от 1 до N. Количество различных перестановок порядка N равно $P_N = N!$

Пусть у нас есть упорядоченное множество из N элементов. Перестановка задает преобразование этого множества. А именно, она говорит, что на i место нужно поставить ai элемент множества, где $a_i$ - i-тый элемент перестановки.

*Обратной перестановкой* к перестановке $\pi$ называется такая перестановка $\pi^{-1}$, что $\pi\pi^{-1} = \pi^{-1}\pi = \varepsilon$, где $\varepsilon$ – тождественная перестановка.

*Степенью перестановки* называется минимальное натуральное число k такое, что $\pi^k = \varepsilon$.

**Входные данные**

В первой строке входного файла INPUT.TXT записано число $0 < N \le 100$ - порядок перестановки. Во второй строке записана сама перестановка.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите степень данной перестановки. Гарантируется, что ответ не превышает $10^9$.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3<br>2 3 1 | 3 |

## 161. Восстановление перестановки

*(Время: 1 сек. Память: 16 Мб Сложность: 44%)*

*Перестановкой* из N элементов называется упорядоченный набор из N различных чисел от 1 до N.

Пусть у нас есть упорядоченное множество из N элементов. Перестановка задает преобразование этого множества. А именно, она говорит, что на i место нужно поставить ai элемент множества, где $a_i$ - i-тый элемент перестановки.

Пусть дана перестановка $\pi$. Обозначим $\varphi[i]$ - количество таких j, что $\pi[j] > \pi[i]$, а $j < i$. $\varphi$ называется *таблицей инверсий* перестановки $\pi$.

Требуется по данной таблице инверсий восстановить перестановку.

**Входные данные**

В первой строке входного файла INPUT.TXT записано число $0 < N \le 2000$ - порядок перестановки. Во второй строке записана таблица инверсий.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите искомую перестановку.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3<br>0 0 2 | 2 3 1 |

## 162. Манхэттэнский полицейский

*(Время: 1 сек. Память: 16 Мб Сложность: 56%)*

Недавно Билл устроился на работу полицейским. Теперь ему предстоит каждый вечер обходить свой участок, который представляет собой прямоугольник, состоящий из N x M кварталов. Каждый квартал имеет вид квадрата размером 100 x 100 метров, кварталы отделены друг от друга прямыми улицами.

Таким образом, через участок Билла проходит N+1 улица, идущая с запада на восток, и M+1 улица, идущая с севера на юг. Перекрестки разбивают улицы на (N+1)*M + (M+1)*N отрезков, каждый из которых имеет длину 100 метров.

Совершая обход, Билл выходит из полицейского управления, расположенного около юго-западного угла его участка, обходит свой участок и возвращается в управление. Во время обхода Билл должен пройти по каждому отрезку улицы на территории своего участка как минимум один раз. Известно, что во время обхода Билл проходит отрезок длиной 100 метров за одну минуту. Выясните, какое минимальное число минут потребуется Биллу, чтобы совершить обход.

**Входные данные**

Входной файл INPUT.TXT содержит натуральные числа N и M, не превышающие 10 000.

**Выходные данные**

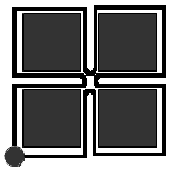
В выходной файл OUTPUT.TXT выведите минимальное время, за которое Билл может совершить обход.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 1 1 | 4 |
| 2 | 2 2 | 16 |
| 3 | 4 3 | 38 |

**Пояснение**

Один из возможных оптимальных путей для Билла во втором примере показан на рисунке:



## 163. Уравнение для 5 класса!

*(Время: 1 сек. Память: 16 Мб Сложность: 25%)*

Уравнение для пятиклассников представляет собой строку длиной 5 символов. Второй символ строки является либо знаком «+» (плюс) либо «-» (минус), четвёртый символ – знак «=» (равно). Из первого, третьего и пятого символов ровно два являются цифрами из диапазона от 0 до 9, и один – буквой x, обозначающей неизвестное.

Требуется написать программу, которая решить данное уравнение относительно x.

**Входные данные**

Входной файл INPUT.TXT состоит из одной строки, в которой записано уравнение.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите целое число – значение x.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | x+5=7 | 2 |
| 2 | 3-x=9 | -6 |

## 164. Счастливый билет - 2

*(Время: 1 сек. Память: 16 Мб Сложность: 26%)*

Билет называется счастливым, если его можно разрезать прямой линией на две части таким образом, что оказавшиеся на них числа имеют одинаковые цифровые корни. Чтобы вычислить цифровой корень числа, его цифры складывают, если в результате получится число большее или равное 10, то цифры складывают снова и так далее, пока не получится число от 0 до 9 – это и есть цифровой корень. Например, билет с номером 0015420 является счастливым, так как разрезав его на части с числами 0015 и 420 имеем у этих чисел одинаковые цифровые корни.

Требуется написать программу, которая определит, является ли счастливым билет с заданным номером.

**Входные данные**

Входной файл INPUT.TXT содержит номер счастливого билета. Номер может начинаться с нулей и содержит не более 100 цифр.

**Выходные данные**

В выходной текстовый файл OUTPUT.TXT выведите «YES», если билет счастливый и «NO» иначе.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 0015420 | YES |
| 2 | 00100 | NO |

## 165. Только вправо или вниз

*(Время: 1 сек. Память: 16 Мб Сложность: 32%)*

Игровое поле N x M заполняется целыми числами, одно неотрицательное целое число в каждой клетке. Цель игры состоит в том, чтобы пройти по любому разрешенному пути от верхнего левого угла до правого нижнего. Целое число в каждой клетке указывает, какой длины шаг должен быть из текущей клетки. Все шаги могут быть или направо или вниз. Если в результате какого-либо шага игрок покидает пределы поля, такой шаг запрещается.

На рис. 1 приведен пример игрового поля 4 x 4, где сплошная окружность показывает положение начала, а пунктирная окружность – цель. Рис. 2 показывает три возможных пути от начала до цели для рассматриваемого примера игрового поля, с удаленными промежуточными числами.

| | | | |
|---|---|---|---|
| 2 | 1 | 1 | 2 |
| 3 | 2 | 1 | 44 |
| 3 | 1 | 1 | 0 |

Рис. 1.

| | | | |
|---|---|---|---|
| 2 | | | |
| | | | |
| 3 | | | 0 |

| | | | |
|---|---|---|---|
| 2 | | 1 | |
| | | 1 | |
| | | 1 | 0 |

| | | | |
|---|---|---|---|
| 2 | | 1 | 2 |
| | | | |
| | | | 0 |

Рис. 2.

Требуется написать программу, которая определит число различных вариантов путей от верхнего левого угла до правого нижнего.

**Входные данные**

Входной файл INPUT.TXT содержит в первой строке размеры поля N ($1 \le N \le 70$) и M ($1 \le M \le 70$). В последующих N строках входного файла, каждая из которых описывает отдельную строку игрового поля, записаны через пробел по M целых чисел – длины шагов из клеток данной строки.

**Выходные данные**

Выходной файл OUTPUT.TXT должен содержать одно число – число различных вариантов путей от верхнего левого угла до правого нижнего. Для каждого поля будет менее чем $2^{31}$ различных путей.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3 4<br>2 1 1 2<br>3 2 1 44<br>3 1 1 0 | 3 |

## 166. Сообщество роботов

*(Время: 1 сек. Память: 16 Мб Сложность: 30%)*

Сообщество роботов живет по следующим законам:

- один раз в начале года они объединяются в группы по три или пять роботов;
- за год группа из трех роботов собирает 5 новых, а группа из 5 роботов – 9 новых;
- роботы объединяются так, чтобы собрать за год наибольшее количество новых роботов;
- каждый робот живет ровно три года после сборки.

В начале первого года было K роботов и все они были только что собраны.

Требуется написать программу, которая найдет количество роботов в начале N-го года.

**Входные данные**

Входной файл INPUT.TXT содержит записанные через пробел числа K ($1 \le K \le 100$) и N ($1 \le N \le 100$).

**Выходные данные**

Выходной файл OUTPUT.TXT должен содержать одно число – количество роботов в начале N-го года. Количество роботов не превышает $2^{31}$.

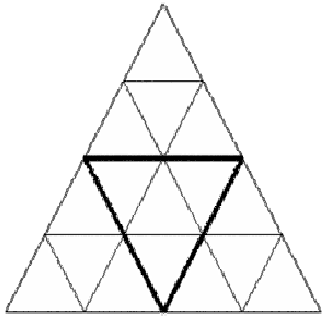
**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3 2 | 8 |
| 2 | 8 2 | 22 |

## 167. Количество треугольников

*(Время: 1 сек. Память: 16 Мб Сложность: 51%)*

Рассмотрим фигуру, аналогичную показанной на рисунке (большой равносторонний треугольник, составленный из маленьких равносторонних треугольников). На рисунке приведена фигура, состоящая из 4-х уровней треугольников.



Требуется написать программу, которая будет определять, сколько всего в ней треугольников (необходимо учитывать не только «маленькие» треугольники, а вообще все треугольники – в частности, треугольник, выделенный жирным, а также вся фигура, являются интересующими нас треугольниками).

**Входные данные**

Входной файл INPUT.TXT содержит одно число N – количество уровней в фигуре ($1 \le N \le 10^5$).

**Выходные данные**

Выходной файл OUTPUT.TXT должен содержать одно число – количество треугольников в такой фигуре.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 2 | 5 |
| 3 | 4 | 27 |

## 168. Натуральный ряд чисел

*(Время: 1 сек. Память: 16 Мб Сложность: 28%)*

Натуральные числа записаны в строку без каких-либо разделителей. Начало этой строки имеет вид 12345678910111213141516171819202l… .

Требуется написать программу, которая определит первое вхождение десятичной записи заданного числа N в этой строке.

**Входные данные**

Входной файл INPUT.TXT содержит заданное число N ($1 \le N \le 10^4$).

**Выходные данные**

Выходной файл OUTPUT.TXT должен содержать одно число – позицию, начиная с которой в строке записано первое вхождение заданного числа. Нумерация позиций начинается с единицы.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 45 | 4 |
| 2 | 101 | 10 |
| 3 | 142 | 73 |

## 169. Магазин

*(Время: 1 сек. Память: 16 Мб Сложность: 34%)*

На расстоянии N шагов от магазина стоит человек. Каждую минуту он выбирает, куда сделать шаг: к магазину или в противоположном направлении.

Требуется написать программу, которая определит, сколькими способами он может попасть в магазин, пройдя ровно K шагов.

**Входные данные**

Входной файл INPUT.TXT содержит в числа n и k, записанные через пробел. Известно, что $1 \le N \le K \le 37$.

**Выходные данные**

Выходной файл OUTPUT.TXT должен содержать одно число – количество способов попадания в магазин.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 2 4 | 2 |
| 2 | 5 5 | 1 |

## 170. Разложение числа

*(Время: 1 сек. Память: 16 Мб Сложность: 35%)*

Любое натуральное число можно представить в виде суммы нескольких последовательных натуральных чисел. Например, число 25 можно представить в виде суммы из одного (25), двух (12+13) или пяти (3+4+5+6+7) чисел.

Требуется написать программу, которая определит максимальное количество чисел в таком разложении.

**Входные данные**

Входной файл INPUT.TXT содержит одно натуральное число N ($1 \le N \le 10^9$).

**Выходные данные**

Выходной файл OUTPUT.TXT должен содержать одно натуральное число – максимальное количество чисел в разложении числа N на сумму последовательных натуральных чисел.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 2 | 1 |
| 2 | 5 | 2 |
| 3 | 25 | 5 |

## 171. Количество делителей

*(Время: 1 сек. Память: 16 Мб Сложность: 50%)*

Пусть x – натуральное число. Назовем y его делителем, если $1 \le y \le x$ и остаток от деления x на y равен нулю.

Задано число x. Найдите количество его делителей.

**Входные данные**

Входной файл INPUT.TXT содержит заданное число x ($1 \le x \le 10^{18}$). Все простые делители числа x не превосходят 1000.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите ответ на задачу.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 12 | 6 |
| 2 | 239 | 2 |

## 172. Деление с остатком

*(Время: 1 сек. Память: 16 Мб Сложность: 39%)*

Заданы два числа: N и K. Необходимо найти остаток от деления N на K.

**Входные данные**

Входной файл INPUT.TXT содержит два целых числа: N и K ($1 \le N \le 10^{100}$, $1 \le K \le 10^{9}$).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите остаток от деления N на K.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 239 16 | 4 |
| 2 | 463874674764573128934748392 7 6784789 | 1001783 |

**Разбор**

Решение данной задачи похоже на решение задачи «A div B». Здесь следует учесть, что делимое – достаточно большое число и в процессе вычисления текущее значение может превосходить максимально возможное для 4-байтного целого, поэтому нужно использовать другие типы (например, int64 или __int64 в паскале).

Алгоритм, реализующий данную задачу может, быть записан в следующем виде:

```
const maxsize=101;
int a[maxsize], b;
int64 x;

readlong(a);
read(b);

x=0; k=0;
for i=a[0]..1{
  x = x*10+a[i];
  if(x < b and k=0 and i > 1) continue;
```

```
    k=1;
    x = x mod b;
  }

  write(x);
```

## 173. Число - палиндром

*(Время: 1 сек. Память: 16 Мб Сложность: 29%)*

Напомним, что палиндромом называется строка, одинаково читающаяся с обеих сторон. Например, строка «ABBA» является палиндромом, а строка «ABC» – нет.

Необходимо определить, в каких системах счисления с основанием от 2 до 36 представление заданного числа N является палиндромом.

В системах счисления с основанием большим 10 в качестве цифр используются буквы латинского алфавита: A, B, ... , Z. Например, $A_{11} = 10_{10}$, $Z_{36} = 35_{10}$.

**Входные данные**

Входной файл INPUT.TXT содержит заданное число N в десятичной системе счисления ($1 \le N \le 10^9$).

**Выходные данные**

Если соответствующее основание системы счисления определяется единственным образом, то выведите в первой строке выходного файла OUTPUT.TXT слово «unique», если оно не единственно – выведите в первой строке выходного файла слово «multiple». Если же такого основания системы счисления не существует – выведите в первой строке выходного файла слово «none».

В случае существования хотя бы одного требуемого основания системы счисления выведите через пробел в возрастающем порядке во второй строке выходного файла все основания системы счисления, удовлетворяющие требованиям.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 123 | unique<br>6 |
| 2 | 111 | multiple<br>6 10 36 |
| 3 | 102892748 | none |

## 174. Свадьба

*(Время: 1 сек. Память: 16 Мб Сложность: 32%)*

Одна предприимчивая и очень симпатичная дамочка с прелестнейшим именем Горгона решила заработать себе денег на роскошную жизнь. N молодых людей так влюблены в нее, что предложили руку и сердце. К несчастью для них, Горгона видит в них только мешок с деньгами. Она планирует выйти замуж и почти сразу же развестись с некоторыми из молодых людей ради денежной выгоды. Все, что ей нужно, это подзаработать как можно больше денег (и уж, конечно, остаться незамужней). По законам этой прекрасной страны при разводе каждый из супругов получает половину всего имущества.

Вы планируете опубликовать статью, в которой опишете всю подлость и меркантильность этой особы. Для того чтобы статья получилась особенно красочной, нужно указать максимальную сумму денег, которую сможет получить Горгона.

**Входные данные**

В первой строке входного файла INPUT.TXT записано целое число N – количество молодых людей, без памяти влюбленных в Горгону ($1 < N < 40$). Далее следует N чисел – сумма денег на счету каждого молодого человека. В последней строке записано целое число A – сумма денег на счету Горгоны. Суммы денег на счету – целые неотрицательные числа, не превосходящие $10^9$.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите единственное число – максимальную сумму денег, которой сможет обладать Горгона после своей махинации. Ответ выводите с точностью до шести знаков ровно в формате без мантиссы.

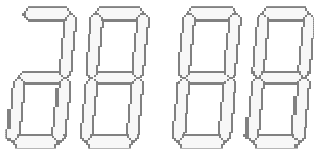
**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 2<br>5 10<br>5 | 7.500000 |
| 2 | 3<br>13 2<br>0 | 2.125000 |

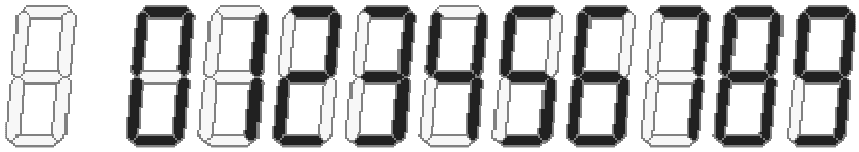
## 175. Наручные часы

*(Время: 1 сек. Память: 16 Мб Сложность: 37%)*

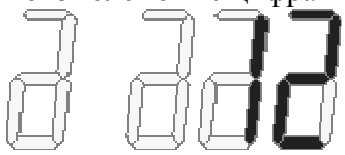
Вы приобрели новые электронные наручные часы с жидкокристаллическим дисплеем. Дисплей отображает часы и минуты с помощью четырех элементов, каждый из которых отображает одну цифру.



Три из них состоят из семи полосок, каждая из которых может быть либо белой (неотличимой от фона), либо черной. Вид такого элемента и отображаемые им цифры показаны на рисунке:



Четвертый элемент предназначен для отображения старшей цифры часа. Если она равна нулю, то элемент полностью неактивен (все полоски белые), иначе показывается соответствующая цифра. Вот как выглядит этот элемент с цифрами:



Вам хочется проверить: все ли в порядке с новым приобретением, а именно, нет ли таких полосок в каком-либо из элементов, которые либо всегда белые, либо всегда черные. Вы хотите начать проверку в некоторое начальное время. Требуется определить, сколько Вам потребуется минут для убеждения в исправности часов.

**Входные данные**

В первой строке входного файла INPUT.TXT находится время начала проверки в формате HH:MM ($00 \le HH \le 23$, $00 \le MM \le 59$). Часы и минуты записаны с лидирующими нулями, если таковые имеются.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите минимальное число минут, необходимое для проверки Ваших часов, если она началась в заданное время.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 00:00 | 1200 |
| 2 | 02:39 | 1041 |

## 176. Скобочки

*(Время: 1 сек. Память: 16 Мб Сложность: 69%)*

Строка, состоящая из символов «(» и «)», называется скобочной последовательностью. Скобочная последовательность называется правильной, если она может быть получена из некоторого корректного арифметического выражения удалением всех символов, кроме скобок. Например, правильная скобочная последовательность «(())()» может быть получена из выражения «(2-(3+4)*6)*(1+1)».

Глубиной правильной скобочной последовательности называется максимальная разность между количеством открывающихся и закрывающихся скобок в префиксе последовательности (префиксом строки S называется строка, которую можно получить из S удалением некоторого количества последних символов, например, префиксами строки «ABCAB» являются строки «», «A», «AB», «ABC», «ABCA» и «ABCAB»). Например, глубина последовательности «()()(())» равна двум, т.к. префикс «()()((» имеет 4 открывающиеся и 2 закрывающиеся скобки.

Требуется написать программу, определяющую по заданным значениям N и K количество правильных скобочных последовательностей с N открывающимися скобками, которые имеют глубину, равную K.

**Входные данные**

Входной файл INPUT.TXT содержит в одной строке целые числа N и K ($1 \le K \le N \le 50$), разделенные пробелом.

**Выходные данные**

Выходной файл OUTPUT.TXT должен содержать одно число – количество правильных скобочных последовательностей с n открывающимися скобками, которые имеют глубину k.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3 2 | 3 |
| 2 | 37 23 | 203685956218528 |

## 177. Склад

*(Время: 1 сек. Память: 16 Мб Сложность: 48%)*

На роботизированном складе имеется n отсеков, в которые робот может размещать грузы. Отсек с номером i имеет вместимость $c_i$. Груз с номером i имеет размер $s_i$, поступает на склад в момент времени $a_i$ и забирается со склада в момент времени $d_i$. Вместимость отсека и размер груза имеют одну и ту же размерность. Если в отсеке с вместимостью c находится несколько грузов с суммарным размером d, то свободное место в этом отсеке равно c – d.

Когда груз с номером i поступает на склад, робот сначала пытается найти отсек, в котором достаточно свободного места для размещения этого груза. Если отсеков, в которых достаточно свободного места, несколько, то робот помещает груз в тот из них, в котором свободного места меньше. Если и таких отсеков несколько, то робот выбирает отсек с минимальным номером.

Если отсеков с достаточным количеством свободного места нет, робот пытается переместить грузы, уже расположенные в отсеках. Для этого он пытается найти такой отсек и такой груз в нем, что перемещение его в другой отсек обеспечивает достаточное количество свободного места для размещения поступившего груза. Если таких вариантов перемещения грузов несколько, то выбирается тот вариант, в котором потребуется перемещение груза с минимальным размером. Если и таких вариантов несколько, то выбирается вариант перемещения, при котором в отсеке, из которого перемещается груз, свободное место после перемещения этого груза будет минимально, а при прочих равных условиях – тот вариант, при котором в отсеке, куда осуществляется перемещение, свободное место после этого перемещения будет также минимально. Если и после этого остается более одного варианта, то выбирается тот вариант, при котором номер перемещаемого груза минимален и номер отсека, в который он перемещается, – также минимален. Если варианта с перемещением одного груза найти не удалось, то груз не принимается на склад.

Требуется написать программу, которая по списку грузов, поступающих для размещения на складе, выводит последовательность действий, выполняемых роботом.

**Входные данные**

Первая строка входного файла содержит два целых числа: n – количество отсеков, и m – количество грузов ($1 \le n \le 10$, $1 \le m \le 100$). Вторая строка содержит n целых чисел $c_i$, определяющих вместимости отсеков ($1 \le c_i \le 10^9$). Последующие m строк описывают грузы: каждый груз описывается тремя целыми числами: своим размером $s_i$, временем поступления на склад $a_i$ и временем, когда его забирают со склада $d_i$ ($1 \le s_i \le 10^9$, $1 \le a_i < d_i \le 1000$, все времена во входном файле различны, грузы упорядочены по возрастанию времени поступления на склад). Все числа в строках разделены пробелом.

**Выходные данные**

Выведите последовательность действий робота в том порядке, в котором они выполняются. Следуйте формату выходного файла, приведенного в примере. Возможны следующие сообщения:

- put cargo X to cell Y – разместить груз с номером X в отсеке с номером Y;
- move cargo X from cell Y to cell Z – переместить груз с номером X из отсека с номером Y в отсек с номером Z;
- take cargo X from cell Y – взять груз с номером X из отсека с номером Y.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3 5<br>3 2 10<br>1 1 6<br>3 2 8<br>9 3 5<br>2 4 9<br>12 7 10 | put cargo 1 to cell 2<br>put cargo 2 to cell 1<br>put cargo 3 to cell 3<br>move cargo 1 from cell 2 to cell 3<br>put cargo 4 to cell 2<br>take cargo 3 from cell 3<br>take cargo 1 from cell 3<br>cargo 5 cannot be stored<br>take cargo 2 from cell 1<br>take cargo 4 from cell 2 |

## 178. Преобразование последовательности

*(Время: 1 сек. Память: 16 Мб Сложность: 42%)*

Задана последовательность, содержащая n целых чисел. Необходимо найти число, которое встречается в этой последовательности наибольшее количество раз, а если таких чисел несколько, то найти минимальное из них, и после этого переместить все такие числа в конец заданной последовательности. Порядок расположения остальных чисел должен остаться без изменения.

Например, последовательность 1, 2, 3, 2, 3, 1, 2 после преобразования должна превратиться в последовательность 1, 3, 3, 1, 2, 2, 2.

Требуется написать программу, которая решает данную задачу.

**Входные данные**

Первая строка входного файла INPUT.TXT содержит число n – количество чисел во входной последовательности ($3 \le n \le 200000$). Следующая строка содержит входную последовательность, состоящую из n целых чисел, не превышающих по модулю $10^9$. Все числа в строке разделены пробелом.

**Выходные данные**

В выходной файл OUTPUT.TXT выводится последовательность чисел, которая получается в результате названного преобразования. Все числа в последовательности должны быть разделены пробелом.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 7<br>1 2 3 2 3 1 2 | 1 3 3 1 2 2 2 |

## 179. Последовательность

*(Время: 1 сек. Память: 16 Мб Сложность: 47%)*

Рассмотрим числовую последовательность, первоначально состоящую из двух единиц: 1, 1. Далее на каждом последующем шаге будем вставлять между соседними элементами их сумму. В примере добавляемые элементы выделены:

| Номер шага | Последовательность |
|---|---|
| 0 | 1, 1 |
| 1 | 1, **2**, 1 |
| 2 | 1, **3**, 2, **3**, 1 |
| 3 | 1, **4**, 3, **5**, 2, **5**, 3, **4**, 1 |

Требуется написать программу, которая подсчитает сумму членов последовательности, построенной за K шагов.

**Входные данные**

Входной файл INPUT.TXT содержит одно натуральное число K ($0 \le K \le 100$) – номер последнего шага.

**Выходные данные**

Выходной файл OUTPUT.TXT должен содержать одно натуральное число – сумму элементов последовательности, построенной за K шагов.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3 | 28 |
| 2 | 10 | 59050 |

## 180. Счастливая страница

*(Время: 1 сек. Память: 16 Мб Сложность: 46%)*

В новом выпуске Большой Галактической Энциклопедии N страниц. Петя считает страницу счастливой, если произведение цифр, входящих в ее номер, равно K. Например, если N=100, то для K=42 есть счастливая страница (например, с номером 76), а для K=128 счастливой страницы нет.

Требуется написать программу, которая поможет Пете определить, есть ли счастливые страницы в новом выпуске энциклопедии.

**Входные данные**

Входной текстовый файл INPUT.TXT содержит числа N ($1 \le N \le 10^9$) и K ($1 \le K \le 10^9$), записанные через пробел.

**Выходные данные**

Выходной текстовый файл OUTPUT.TXT должен содержать «YES», если счастливые страницы есть, и «NO» иначе.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 100 42 | YES |
| 2 | 100 128 | NO |

## 181. Космический мусорщик

*(Время: 1 сек. Память: 16 Мб Сложность: 52%)*

В околоземном космическом пространстве накопилось много мусора, поэтому ученые сконструировали специальный аппарат ловушку для космического мусора. Аппарат должен движется по достаточно сложной траектории, сжигая по пути мусор. Ловушка может передвигаться в пространстве по 6 направлениям: на север (N), на юг (S), на запад (W), на восток (E), вверх (U) и вниз (D). Движением ловушки управляет процессор. Программа движения задается шестью правилами движения, которые соответствуют каждому из указанных направлений. Каждое такое правило представляет собой строку символов из множества {N, S, W, E, U, D}.

Команда ловушки состоит из символа направления и целого положительного числа M. Если параметр больше 1, то ловушка перемещается на один метр в направлении, которое указано в команде, а затем последовательно выполняет команды, заданные правилом для данного направления, с параметром меньше на 1. Если же параметр равен 1, то просто перемещается на один метр в указанном направлении.

Пусть, например, заданы правила, отраженные в таблице. Тогда при выполнении команды S(3) мусорщик сначала переместится на 1 метр в направлении S, а потом выполнит последовательно команды N(2), U(2), S(2), D(2), D(2), U(2), S(2), E(2).

| Направление | Правило |
|---|---|
| N | N |
| S | NUSDDUSE |
| W | UEWWD |
| E | |
| U | U |
| D | WED |

Если далее проанализировать действия мусорщика, получим, что в целом он совершит ровно 34 перемещения.

**Входные данные**

Первые шесть строк входного файла INPUT.TXT задают правила для команд с направлением N, S, W, E, U и D соответственно. Каждая строка содержит не более 100 символов (и может быть пустой). Следующая строка содержит команду ловушки: сначала символ из множества {N, S, W, E, U, D}, затем пробел и параметр команды – целое положительное число, не превышающее 100.

**Выходные данные**

Выведите в выходной файл OUTPUT.TXT единственное число – количество перемещений, которое совершит ловушка. Гарантируется, что ответ не превышает $10^9$.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | N<br>NUSDDUSE<br>UEWWD<br><br>U<br>WED<br>S 3 | 34 |

## 182. Прямоугольник - 2

*(Время: 1 сек. Память: 16 Мб Сложность: 27%)*

Заданы координаты трех вершин прямоугольника. Необходимо определить координаты четвертой вершины.

**Входные данные**

Во входном файле INPUT.TXT записаны через пробел координаты трех вершин прямоугольника в произвольном порядке в формате $x_1$ $y_1$ $x_2$ $y_2$ $x_3$ $y_3$. Все числа целые, не превосходящие 1000 по абсолютной величине.

**Выходные данные**

В выходной файл OUTPUT.TXT нужно вывести через пробел координаты четвертой вершины прямоугольника.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 0 3 0 0 5 0 | 5 3 |
| 2 | 1 4 8 3 7 6 | 2 1 |

## 183. Энты

*(Время: 1 сек. Память: 16 Мб Сложность: 39%)*

Энты были созданы в Первоначальную эпоху вместе с другими обитателями Средиземья. Эльфийские легенды гласят, что когда Варда зажгла звёзды и пробудились Эльфы, вместе с ними пробудились и Энты в Великих Лесах Арды.

Когда Энты пришли в Арду, они ещё не умели говорить – этому искусству их обучали Эльфы, и Энтам это ужасно нравилось. Им доставляло удовольствие изучать разные языки, даже щебетание Людей.

Эльфы выработали хорошую технику обучения энтов своему языку. Первый энт, которого обучили эльфы, выучил всего два слова – «tancave» (да) и «la» (нет). Обученный энт выбрал одного старого и одного молодого энта, не умеющих говорить, и обучил их всем словам, которые знал сам. Затем обучение этих двух энтов продолжили сами эльфы. Каждый обучившийся у эльфов энт снова выбирал из неговорящих сородичей одного старого и одного молодого, обучал их всем словам, которые знал, передавал эльфам и так далее.

Выяснилось, что более молодые энты выучивали у эльфов ещё ровно столько же слов, сколько они узнали от обучавшего их энта. А вот более старые, уже склонные к одеревенению энты, пополняли свой запас всего лишь одним словом. После обучения у эльфов энты до конца света уже не могли выучить ни одного нового слова.

Общее число энтов в Средиземье больше, чем вы думаете. Интересно, а сколько из них знают ровно 150 квенийских слов? Похожую задачу вам предстоит решить.

**Входные данные**

Входной файл INPUT.TXT содержит натуральные числа K и P ($K \le 10^6$; $1 \le P \le 10^9$), записанные через пробел.

**Выходные данные**

Мы понимаем, что число энтов, знающих в точности K слов, может быть слишком велико, поэтому просим вывести в выходной файл OUTPUT.TXT лишь количество энтов, знающих ровно K слов, по модулю P.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 4 10 | 2 |
| 2 | 8 10 | 5 |
| 3 | 360 1000 | 179 |

## 184. Рабочее время

*(Время: 1 сек. Память: 16 Мб Сложность: 26%)*

Иван Иванович – очень ответственный, но очень рассеянный человек. Поэтому когда он начинает очередное дело, он на отдельном листочке пишет дату и время начала (например, 29.01. 10:30), а когда заканчивает, то так же на отдельном листочке – дату и время окончания (например, 02.02. 12:15). Листочки аккуратно укладываются в стопку один на другой. А так как одновременно Иван Иванович может заниматься только одним делом, то листочки однозначно упорядочены в стопке: листок начала какого-то дела, листок окончания этого дела, листок начала, листок окончания… и т.д. Дело начинается в начале минуты, указанной в листочке начала этого дела, а заканчивается в конце минуты, указанной на листочке окончания. Иван Иванович ходит на работу каждый день и его рабочий день продолжается с 10:00 до 18:00. Таким образом, пара листочков «18.11. 15:13» – «20.11. 10:27» была написана при начале и окончании дела длительностью 11ч.15м.

Однажды в конце декабря уборщица Дуся нечаянно уронила эту стопку на пол и, не зная важной закономерности их укладки, собрала листочки обратно в каком-то произвольном порядке. Иван Иванович обнаружил этот прискорбный факт только 31 декабря, когда ему надо было произвести учет своего рабочего времени за год. Год был невисокосный.

Помогите Ивану Ивановичу найти его суммарные затраты времени за год.

**Входные данные**

Входной файл INPUT.TXT содержит в первой строке число листочков K, (K – четное число, не большее 5000). Далее записаны K строк с данными на листочках в формате DD.MM. hh:mm, где DD – число, MM – номер месяца, hh – часы и mm – минуты.

**Выходные данные**

Выходной файл OUTPUT.TXT должен содержать запись вида h:mm – количество часов и минут, отработанных Иваном Ивановичем. При этом число $h \geq 0$ выводится без ведущих нулей, а число $0 \leq mm \leq 59$ выводится с ведущими нулями.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 4<br>15.01. 17:00<br>16.01. 12:00<br>11.02. 14:00<br>30.01. 10:00 | 103:02 |

## 185. Скачки

*(Время: 1 сек. Память: 16 Мб Сложность: 32%)*

Иван Иванович любит ходить на скачки, надеясь на них заработать кругленькую сумму. Ему приглянулась лошадь с номером K, и он решил проверить, сможет ли она выиграть у всех остальных лошадей. Иван Иванович раздобыл информацию, в которой для некоторых пар лошадей сообщается, какая из этих лошадей быстрее. Также он узнал, что у всех лошадей разные скорости.

Требуется написать программу, которая поможет Ивану Ивановичу точно определить может ли выиграть выбранная им лошадь.

**Входные данные**

Входной файл INPUT.TXT содержит в первой строке два целых числа N ($1 \leq N \leq 100$) и K ($1 \leq K \leq N$), где N – количество лошадей, принимающих участие в скачках, K – номер лошади, на которую хочет сделать ставку Иван Иванович. Следующие строки содержат по два числа X и Y ($1 \leq X, Y \leq N$), обозначающие, что лошадь с номером X быстрее лошади с номером Y. Пары X и Y не повторяются. Набор данных завершается строкой, содержащей единственный ноль. Эту строку обрабатывать не надо.

Гарантируется, что информация, раздобытая Иваном Ивановичем, корректна.

**Выходные данные**

Выходной файл OUTPUT.TXT должен содержать слово «Yes», если Иван Иванович уверен в своем выигрыше и «No» в противном случае.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3 1<br>1 2<br>1 3<br>0 | Yes |
| 2 | 3 2<br>2 3<br>0 | No |
| 3 | 4 2<br>3 1<br>2 3<br>0 | No |

## 186. Субботник

*(Время: 1 сек. Память: 16 Мб Сложность: 44%)*

В этом году Иван Иванович решил отметить приход осени субботником, чтобы убрать весь мусор во дворе дома номер 31 по улице Осенней. На субботник он пригласил N знакомых старушек, живущих в том же самом доме. Однако в самом начале мероприятия выяснилось, что по одиночке старушки работают плохо, так как им хочется во время работы еще и поговорить друг с другом.

Иван Иванович подумал и принял волевое решение разбить старушек на группы так, чтобы в каждой группе было не менее 2 старушек. Старушки отличаются друг от друга уровнем разговорчивости, и если в одну группу попадут две старушки, у одной из которых маленький уровень разговорчивости, а у второй – большой, то они не могут поговорить друг с другом и работа будет стопориться.

Назовем разговорчивостью группы разность между максимальным и минимальным уровнями разговорчивости старушек в группе. Например, если уровни разговорчивости старушек в группе равны 7, 3 и 11, то разговорчивость группы равна 11 - 3 = 8. Разговорчивостью разбиения старушек на группы назовем максимальную из разговорчивостей групп, входящих в разбиение.

Требуется написать программу, которая поможет Ивану Ивановичу найти разбиение старушек на группы, разговорчивость которого минимальна.

**Входные данные**

Входной файл INPUT.TXT содержит в первой строке число N ($2 \le N \le 1000$) – количество старушек. Во второй строке записано N чисел от 1 до $10^9$ – разговорчивости старушек.

**Выходные данные**

Выходной текстовый файл OUTPUT.TXT должен содержать одно целое число, равное минимально возможной разговорчивости разбиения старушек на группы.

**Примеры**

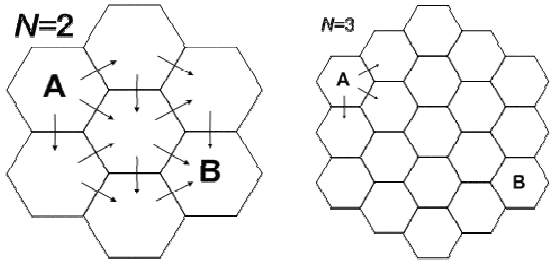
| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 2<br>1 1000000000 | 999999999 |
| 2 | 3<br>1 2 3 | 2 |
| 3 | 8<br>1 10 100 1000 1000 100 10 1 | 0 |
| 4 | 10<br>258 740 156 244 458 680 390 694 844 817 | 102 |

## 187. Пчелка

*(Время: 1 сек. Память: 16 Мб Сложность: 53%)*

Представьте себе пчелиные соты – поле из шестиугольных клеток со стороной N. В верхней левой клетке A находится пчелка. За один ход она может переползти на клетку вниз, на клетку вниз-вправо или на клетку вверх-вправо (вверх и влево пчелка не ползает).

Требуется написать программу, которая найдет количество способов, которыми пчелка может доползти из клетки A в противоположную клетку B.



**Входные данные**

Входной файл INPUT.TXT содержит единственное число N – размеры шестиугольного поля ($2 \le N \le 12$).

**Выходные данные**

Выходной файл OUTPUT.TXT должен содержать единственное целое число – количество способов.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 2 | 11 |
| 2 | 3 | 291 |

## 188. День рождения

*(Время: 1 сек. Память: 16 Мб Сложность: 75%)*

Иван Иванович пригласил на свой день рождения много гостей. Он написал на карточках фамилии всех гостей и разложил эти карточки на столе, полагая, что каждый гость сядет там, где обнаружит карточку со своей фамилией (фамилии у всех гостей различны). Однако гости не обратили внимания на карточки и сели за стол в произвольном порядке. При этом Иван Иванович с удивлением обнаружил, что ни один гость не сел на предназначенное ему место.

Требуется написать программу, которая найдет сколькими способами можно рассадить гостей так, чтобы ни один из них не сидел там, где лежала карточка с его фамилией.

**Входные данные**

Во входном файле INPUT.TXT задано целое число N – количество гостей ($1 \le N \le 100$).

**Выходные данные**

Выходной файл OUTPUT.TXT должен содержать одно целое число – количество способов рассадить гостей.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 1 | 0 |
| 2 | 2 | 1 |
| 3 | 5 | 44 |
| 4 | 20 | 895014631192902121 |

## 189. Перестановка по номеру

*(Время: 1 сек. Память: 16 Мб Сложность: 47%)*

*Перестановкой* из N элементов называется упорядоченный набор из N различных чисел от 1 до N. Количество всех перестановок порядка N равно $P_N = N!$

Требуется найти перестановку по ее номеру в лексикографическом порядке (по алфавиту). Например, для N=3 лексикографический порядок перестановок выглядит следующим образом:

(1,2,3), (1,3,2), (2,1,3), (2,3,1), (3,1,2), (3,2,1).

Таким образом, перестановка (2,3,1) имеет номер 4 в этой последовательности.

**Входные данные**

В первой строке входного файла INPUT.TXT записано число N ($1 \le N \le 12$) – количество элементов в перестановке, во второй - число K ($1 \le K \le N!$) - номер перестановки.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите через пробел N чисел – искомую перестановку.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 1<br>1 | 1 |
| 2 | 3<br>2 | 1 3 2 |

## 190. По размещению!

*(Время: 1 сек. Память: 16 Мб Сложность: 56%)*

*Перестановкой* из N элементов называется упорядоченный набор из N различных чисел от 1 до N.

*Размещением* порядка K называют подмножество элементов некоторой перестановки порядка N. Например, (1, 3) – размещение порядка 2 для перестановки (1, 2, 3) порядка 3.

Требуется по заданному размещению определить его позицию в лексикографическом порядке всех возможных размещений, образованных из всевозможных перестановок порядка N.

Например, лексикографическая последовательность всевозможных размещений для K=2 и N=3 выглядит следующим образом: (1,2), (1,3), (2,1), (2,3), (3,1), (3,2).

Таким образом, перемещение (2,3) имеет номер 4 в этой последовательности.

**Входные данные**

В первой строке входного файла INPUT.TXT находятся числа N и K ($1 \le K \le N \le 12$). Во второй строке записаны K чисел из диапазона от 1 до N – размещение.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите единственное число – номер данного размещения.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3 2<br>3 2 | 6 |
| 2 | 6 4<br>1 3 2 5 | 14 |

## 191. Гладкие числа

*(Время: 1 сек. Память: 16 Мб Сложность: 60%)*

Назовем число *гладким*, если его цифры, начиная со старшего разряда, образуют неубывающую последовательность. Упорядочим все такие числа в возрастающем порядке и присвоим каждому номер.

Вам требуется по номеру N вывести N-ое гладкое число.

**Входные данные**

Во входном файле INPUT.TXT содержится номер N ($1 \le N \le 2147483647$).

**Выходные данные**

Выходной файл OUTPUT.TXT должен содержать искомое N-е гладкое число.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 11 | 12 |
| 3 | 239 | 1135 |

## 192. Следующая перестановка ...

*(Время: 1 сек. Память: 16 Мб Сложность: 46%)*

*Перестановкой* из N элементов называется упорядоченный набор из N различных чисел от 1 до N.

Найдите по заданной перестановке следующую в лексикографическом порядке (будем считать, что за перестановкой (N, N-1, ... , 3, 2, 1) следует тождественная перестановка, то есть (1, 2, 3, ... , N)).

**Входные данные**

В первой строке входного файла INPUT.TXT содержится число N ($1 \le N \le 10^4$). Во второй строке содержится перестановка (последовательность натуральных чисел от 1 до N, разделенных пробелами).

**Выходные данные**

Выходной файл OUTPUT.TXT должен содержать искомую перестановку.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 1<br>1 | 1 |
| 2 | 5<br>2 4 5 3 1 | 2 5 1 3 4 |

## 193. Поиск прямоугольников

*(Время: 0,25 сек. Память: 16 Мб Сложность: 34%)*

На поле NxM клеток (N строк и M столбцов) положили K прямоугольников один поверх другого в случайном порядке. Длины сторон прямоугольников выражаются целым числом клеток. Прямоугольники не выходят за границы поля. Границы прямоугольников совпадают с границами клеток поля.

Получившуюся ситуацию записали в таблицу чисел (каждой клетке поля соответствует клетка таблицы). Если клетка поля не закрыта прямоугольником, то в соответствующую клетку таблицы записали число 0. Если же клетка закрыта одним или несколькими прямоугольниками, то в соответствующую клетку таблицы записали число, соответствующее номеру самого верхнего прямоугольника, закрывающего эту клетку.

Требуется написать программу, которая определит положение и размеры прямоугольников. Гарантируется, что во входных данных содержится информация, которой достаточно для однозначного определения размеров прямоугольников.

**Входные данные**

Входной файл INPUT.TXT содержит в первой строке целые числа N, M, K ($1 \le N \le 200$, $1 \le M \le 200$, $1 \le K \le 255$). Далее следует N строк по M чисел в каждой – содержимое таблицы. Все числа в таблице целые, находятся в диапазоне от 0 до K включительно.

**Выходные данные**

Выходной файл OUTPUT.TXT должен содержать K строк. Каждая строка должна описывать соответствующий ее номеру прямоугольник четырьмя числами X1 Y1 X2 Y2 (X1 и Y1 должны описывать координаты левого нижнего угла прямоугольника, а X2 и Y2 – координаты правого верхнего угла). Числа должны разделяться пробелом.

| | | | | |
|---|---|---|---|---|
| 0 | 2 | 2 | 2 | 2 |
| 0 | 2 | 2 | 2 | 2 |
| 1 | 1 | 2 | 2 | 2 |
| 1 | 1 | 0 | 0 | 0 |

Начало координат расположено в левом нижнем углу таблицы. Таким образом, координаты левого нижнего угла поля – (0,0), правого верхнего – (M,N).

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 4 5 2<br>0 2 2 2 2<br>0 2 2 2 2<br>1 1 2 2 2<br>1 1 0 0 0 | 0 0 2 2<br>1 1 5 4 |

## 194. Фотограф-зануда

*(Время: 1 сек. Память: 16 Мб Сложность: 55%)*

Однажды глава семейства заказал фотографию своей большой семьи, состоящей из N человек, возраст которых 1 год, 2 года, …, N-1 лет и N лет. На фотографии должны присутствовать все родственники, и для этого они должны расположиться в один ряд. Сначала было решено расположить родственников по старшинству, начиная с самого младшего. Но фотограф сказал, что, возможно, на фото это будет выглядеть неестественно. Тогда было решено использовать следующее размещение:

1. слева сидит ребенок возрастом в 1 год,
2. разность возрастов двух соседних родственников не превышает 2 года.

Действительно, на фотографии, таким образом, все будут все равно выглядеть, будто расположенные по старшинству (ведь среди людей возрастом, к примеру, 25 и 27 лет не так легко определить старшего). Способов такой посадки существует, понятно, несколько. Фотограф снял все такие способы. Сколько же фотографий получилось в итоге?

**Входные данные**

Во входном файле INPUT.TXT содержится число N ($1 \le N \le 55$) – количество членов большой семьи.

**Выходные данные**

Выходной файл OUTPUT.TXT должен содержать искомое число фотографий.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 4 | 4 |
| 2 | 7 | 14 |

## 195. Эния

*(Время: 1 сек. Память: 16 Мб Сложность: 3%)*

Неспокойно сейчас на стапелях шестого дока межгалактического порта планеты Торна. Всего через месяц закончится реконструкция малого броненесущего корвета «Эния». И снова этому боевому кораблю и его доблестной команде предстоят тяжелые бои за контроль над плутониевыми рудниками Сибелиуса. Работа не прекращается ни на секунду, лазерные сварочные аппараты работают круглые сутки. От непрерывной работы плавятся шарниры роботов-ремонтников. Но задержаться нельзя ни на секунду.

И вот в этой суматохе обнаруживается, что термозащитные панели корвета вновь требуют срочной обработки сульфидом тория. Известно, что на обработку одного квадратного метра панели требуется 1 нанограмм сульфида. Всего необходимо обработать N прямоугольных панелей размером A на B метров. Вам необходимо как можно скорее подсчитать, сколько всего сульфида необходимо на обработку всех панелей «Энии». И не забудьте, что панели требуют обработки с обеих сторон.

**Входные данные**

Во входном файле INPUT.TXT содержатся 3 целых положительных числа N ($N \le 100$), A ($A \le 100$), B ($B \le 100$).

**Выходные данные**

В выходной файл OUTPUT.TXT нужно вывести единственное число – вес необходимого для обработки сульфида тория в нанограммах.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 5 2 3 | 60 |
| 2 | 14 23 5 | 3220 |

## 196. Спираль

*(Время: 1 сек. Память: 16 Мб Сложность: 38%)*

Требуется совершить обход квадратной матрицы по спирали так, как показано на рисунке справа: заполнение происходит с единицы из левого верхнего угла и заканчивается в центре числом $N^2$, где N – порядок матрицы.

| | | | | |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 |
| 16 | 17 | 18 | 19 | 6 |
| 15 | 24 | 25 | 20 | 7 |
| 14 | 23 | 22 | 21 | 8 |
| 13 | 12 | 11 | 10 | 9 |

**Входные данные**

Во входном файле INPUT.TXT задано натуральное число N – размер квадратной матрицы ($N \le 100$).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите матрицу, заполненную числами от 1 до $N^2$ по спирали, при этом между числами может быть любое количество пробелов.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 5 | 1 2 3 4 5<br>16 17 18 19 6<br>15 24 25 20 7<br>14 23 22 21 8<br>13 12 11 10 9 |

## 197. Змейка

*(Время: 1 сек. Память: 16 Мб Сложность: 40%)*

Требуется заполнить змейкой квадратную матрицу так, как показано на рисунке справа: заполнение происходит с единицы из левого верхнего угла и заканчивается в правом нижнем числом $N^2$, где N – порядок матрицы.

| | | | |
|---|---|---|---|
| 1 | 3 | 4 | 10 |
| 2 | 5 | 9 | 11 |
| 6 | 8 | 12 | 15 |
| 7 | 13 | 14 | 16 |

**Входные данные**

Во входном файле INPUT.TXT задано натуральное число N – размер квадратной матрицы ($N \le 100$).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите матрицу, заполненную числами от 1 до $N^2$ змейкой, при этом между числами может быть любое количество пробелов.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 4 | 1 3 4 10<br>2 5 9 11<br>6 8 12 15<br>7 13 14 16 |

## 198. Система линейных уравнений

*(Время: 1 сек. Память: 16 Мб Сложность: 57%)*

Требуется решить невырожденную систему, состоящую из N линейных уравнений с N неизвестными:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2 \\ \ldots \\ a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n = b_i \\ \ldots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = b_n \end{cases}$$

**Входные данные**

В первой строке входного файла INPUT.TXT задано натуральное число N – ранг системы, далее следуют N строк, каждая из которых состоит из N+1 целых чисел: коэффициенты i-й строки уравнения – N чисел $a_{ij}$ и $b_i$. ($N \le 100$, $|a_{ij}|<10$, $|b_i|<10^4$).

**Выходные данные**

В единственную строку выходного файла OUTPUT.TXT нужно вывести через пробел корни приведенной во входном файле системы линейных уравнений. Гарантируется, что все корни целые и не превосходят значения 10 по абсолютной величине.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 2<br>5 9 62<br>9 -3 54 | 7 3 |

| | | |
|---|---|---|
| 2 | 3<br>7 -9 1 2<br>0 6 5 -15<br>3 -3 2 -9 | 8 5 -9 |

## 199. Римские числа

*(Время: 1 сек. Память: 16 Мб Сложность: 50%)*

Необходимо сократить дробь, записанную в римской системе счисления. Напомним, что в римской записи используются символы M, D, C, L, X, V и I. Приведем таблицу с примерами перевода римских чисел в арабскую систему:

| | | | |
|---|---|---|---|
| I - 1 | VII - 7 | XLVI - 46 | CCCII - 302 |
| II - 2 | VIII - 8 | L - 50 | CDXLI - 441 |
| III - 3 | IX - 9 | LXXV - 75 | ID - 499 |
| IV - 4 | X - 10 | XCII - 92 | D - 500 |
| V - 5 | XVIII - 18 | IC - 99 | DCXCV - 695 |
| VI - 6 | XXXI - 31 | C - 100 | CM - 900 |

**Входные данные**

Во входном файле INPUT.TXT записана дробь в римской системе счисления. Формат записи считается корректным, если запись представляет собой: римское число, деление, римское число (без пробелов), и каждое из чисел находится в диапазоне от 1 до 999.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите сокращенную дробь в римской системе счисления. В тех случаях, когда первое число делится на второе, следует выводить результат в виде только одного римского числа. В том случае, когда во входных данных содержится ошибка, следует вывести ERROR.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | II/IV | I/II |
| 2 | XXIV/VIII | III |
| 3 | 12/16 | ERROR |

## 200. Марсианские факториалы

*(Время: 1 сек. Память: 16 Мб Сложность: 77%)*

В 3141 году очередная экспедиция на Марс обнаружила в одной из пещер таинственные знаки. Они однозначно доказывали существование на Марсе разумных существ. Однако смысл этих таинственных знаков долгое время оставался неизвестным. Недавно один из ученых, профессор Очень-Умный, заметил один интересный факт: всего в надписях, составленных из этих знаков, встречается ровно K различных символов. Более того, все надписи заканчиваются на длинную последовательность одних и тех же символов.

Вывод, который сделал из своих наблюдений профессор, потряс всех ученых Земли. Он предположил, что эти надписи являются записями факториалов различных натуральных чисел в системе счисления с основанием K. А символы в конце – это конечно же нули – ведь, как известно, факториалы больших чисел заканчиваются большим количеством нулей. Например, в нашей десятичной системе счисления факториалы заканчиваются на нули начиная с 5! = 1·2·3·4·5 = 120. А у числа 100! в конце следует 24 нуля в десятичной системе счисления и 48 нулей в системе счисления с основанием 6 – так что у предположения профессора есть разумные основания!

Теперь ученым срочно нужна программа, которая по заданным числам N и K найдет количество нулей в конце записи в системе счисления с основанием K числа N! = 1·2·3·…·(N-1)·N, чтобы они могли проверить свою гипотезу. Вам придется написать им такую программу!

**Входные данные**

Входной файл INPUT.TXT содержит числа N и K, разделенные пробелом ($1 \le N \le 10^9$, $2 \le K \le 1000$).

**Выходные данные**

Выведите в выходной файл OUTPUT.TXT число X – количество нулей в конце записи числа N! в системе счисления с основанием K.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 5 10 | 1 |
| 2 | 100 10 | 24 |
| 3 | 100 6 | 48 |
| 4 | 3 10 | 0 |

## 201. Пакетная обработка процессов

*(Время: 1 сек. Память: 16 Мб Сложность: 48%)*

Для ускорения прохождения «коротких» заданий на ЭВМ выбран пакетный режим работы с квантованием времени процессора. Это значит, что всем заданиям пакета по очереди представляется процессор на одинаковое время 10 с (круговой циклический алгоритм разделения времени). Если в течение этого времени заканчивается выполнение задания, оно покидает систему и освобождает процессор. Если же очередного кванта времени не хватает для завершения задания, оно помещается в конец очереди – пакета. Последнее задание пакета выполняется без прерываний. Пакет считается готовым к вводу в ЭВМ, если в нем содержится K заданий. Новый пакет вводится в ЭВМ после окончания обработки предыдущего. Задания поступают в систему с интервалом времени 60 ± 30 с и характеризуются временем работы процессора 50 ± 45 с.

Требуется смоделировать процесс обработки N заданий и определить время начала и окончания каждого процесса.

**Входные данные**

На первой строке входного файла INPUT.TXT находятся числа N и K – число процессов и количество процессов в пакете ($1 \le N \le 1000$, $1 \le K \le 100$). Гарантируется, что N делится на K. Далее следуют N строк с информацией о времени формирования и необходимое время на выполнение для каждого процесса. Все процессы следуют в порядке возрастания времени ввода их в систему.

**Выходные данные**

Выведите в выходной файл OUTPUT.TXT для каждого процесса в отдельной строке время его старта и время окончания через пробел в формате ЧЧ:ММ:СС.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 6 2<br>00:01:02 63<br>00:02:14 76<br>00:03:16 14<br>00:04:02 19<br>00:04:36 17<br>00:05:45 75 | 00:02:14 00:04:17<br>00:02:24 00:04:33<br>00:04:33 00:04:57<br>00:04:43 00:05:06<br>00:05:45 00:06:12<br>00:05:55 00:07:17 |

## 202. Поиск подстроки

*(Время: 1 сек. Память: 16 Мб Сложность: 44%)*

Найти все вхождения строки T в строке S.

**Входные данные**

В первой строке входного файла INPUT.TXT записана строка S, во второй строке записана строка T. Обе строки состоят только из латинских букв. Длины строк больше 0 и меньше 50 000.

**Выходные данные**

В выходной файл OUTPUT.TXT нужно вывести все вхождения строки T в строку S в порядке возрастания. Нумерация позиций строк начинается с нуля.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | ababbababa<br>aba | 0 5 7 |

## 203. Сдвиг текста

*(Время: 1 сек. Память: 16 Мб Сложность: 51%)*

Мальчик Кирилл написал однажды на листе бумаги строчку, состоящую из больших и маленьких латинских букв, а после этого ушел играть в футбол. Когда он вернулся, то обнаружил, что его друг Дима написал под его строкой еще одну строчку такой же длины. Дима утверждает, что свою строчку он получил циклическим сдвигом строки Кирилла направо на несколько шагов (циклический сдвиг строки abcde на 2 позиции направо даст строку deabc). Однако Дима известен тем, что может случайно ошибиться в большом количестве вычислений, поэтому Кирилл в растерянности – верить ли Диме? Помогите ему!

По данным строкам выведите минимально возможный размер сдвига или -1, если Дима ошибся.

**Входные данные**

Первые две строки входного файла INPUT.TXT содержат строки Кирилла и Димы соответственно. Строки состоят только из латинских символов. Длины строк одинаковы, не превышают 10000 и не равны 0.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите единственное число - ответ на поставленную задачу.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | abcde<br>deabc | 2 |

## 204. Циклическая строка

*(Время: 1 сек. Память: 16 Мб Сложность: 53%)*

Строка S была записана много раз подряд, после чего из получившейся строки взяли подстроку и дали Вам. Ваша задача определить минимально возможную длину исходной строки S.

**Входные данные**

В единственной строке входного файла INPUT.TXT записана строка, которая содержит только латинские буквы, длина строки не превышает 50000 символов.

**Выходные данные**

В выходной файл OUTPUT.TXT нужно вывести одно число - ответ на задачу.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | abababa | 2 |

## 205. Таймер

*(Время: 1 сек. Память: 16 Мб Сложность: 31%)*

Таймер – это часы, которые умеют подавать звуковой сигнал по прошествии некоторого периода времени. Напишите программу, которая определяет, когда должен быть подан звуковой сигнал.

**Входные данные**

В первой строке входного файла INPUT.TXT записано текущее время в формате ЧЧ:ММ:СС (с ведущими нулями). При этом оно удовлетворяет ограничениям: ЧЧ – от 00 до 23, ММ и СС – от 00 до 60.

Во второй строке записан интервал времени, который должен быть измерен. Интервал записывается в формате Ч:М:С (где Ч, М и С – от 0 до $10^9$, без ведущих нулей). Дополнительно если Ч=0 (или Ч=0 и М=0), то они могут быть опущены. Например, 100:60 на самом деле означает 100 минут 60 секунд, что то же самое, что 101:0 или 1:41:0. А 42 обозначает 42 секунды. 100:100:100 – 100 часов, 100 минут, 100 секунд, что то же самое, что 101:41:40.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите в формате ЧЧ:ММ:СС время, во сколько прозвучит звуковой сигнал. При этом если сигнал прозвучит не в текущие сутки, то дальше должна следовать запись +<кол во> days. Например, если сигнал прозвучит на следующий день – то +1 days.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 01:01:01<br>48:0:0 | 01:01:01+2 days |
| 2 | 01:01:01<br>58:119 | 02:01:00 |
| 3 | 23:59:59<br>1 | 00:00:00+1 days |

## 206. Домой на электричках

*(Время: 1 сек. Память: 16 Мб Сложность: 40%)*

Одна из команд-участниц олимпиады решила вернуться домой на электричках. При этом ребята хотят попасть домой как можно раньше. К сожалению, не все электрички идут от города, где проводится олимпиада, до станции, на которой живут ребята. И, что еще более обидно, не все электрички, которые идут мимо их станции, останавливаются на ней (равно как вообще, электрички останавливаются далеко не на всех станциях, мимо которых они идут).

Все станции на линии пронумерованы числами от 1 до N. При этом станция номер 1 находится в городе, где проводится олимпиада, и в момент времени 0 ребята приходят на станцию. Станция, на которую нужно попасть ребятам, имеет номер E.

Напишите программу, которая по данному расписанию движения электричек вычисляет минимальное время, когда ребята могут оказаться дома.

**Входные данные**

Во входном файле INPUT.TXT записаны сначала числа N ($2 \le N \le 100$) и E ($2 \le E \le N$). Затем записано число M ($0 \le M \le 100$), обозначающее число рейсов электричек. Далее идет описание M рейсов электричек. Описание каждого рейса электрички начинается с числа $K_i$ ($2 \le K_i \le N$) – количества станций, на которых она останавливается, а далее следует Ki пар чисел, первое число каждой пары задает номер станции, второе – время, когда электричка останавливается на этой станции (время выражается целым числом из диапазона от 0 до $10^9$). Станции внутри одного рейса упорядочены в порядке возрастания времени. В течение одного рейса электричка все время движется в одном направлении – либо от города, либо к городу.

**Выходные данные**

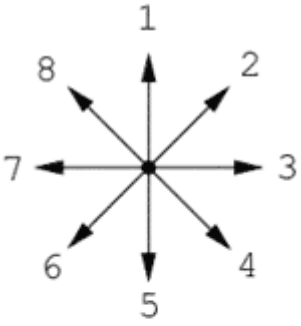
В выходной файл OUTPUT.TXT выведите одно число – минимальное время, когда ребята смогут оказаться на своей станции. Если существующими рейсами электричек они добраться не смогут, выведите –1.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 5 3<br>4<br>2 1 5 2 10<br>2 2 10 4 15<br>4 5 0 4 17 3 20 2 35<br>3 1 2 3 40 4 45 | 20 |

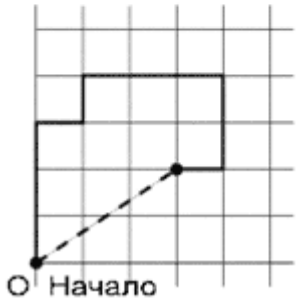
## 207. Клад

*(Время: 1 сек. Память: 16 Мб Сложность: 28%)*




Найти закопанный пиратами клад просто: всё, что для этого нужно – это карта. Как известно, пираты обычно рисуют карты от руки и описывают алгоритм действий. Большая часть таких действий просто сводится к прохождению какого-то количества шагов в одном из восьми направлений (1 – север, 2 – северо-восток, 3 – восток, 4 – юго-восток, 5 – юг, 6 – юго-запад, 7 – запад, 8 – северо-запад) (см. рис). Длина шага в любом направлении равна 1.

Путешествие по такому пути обычно является прекрасным способом посмотреть окрестности, однако в наше время постоянной спешки ни у кого нет времени на это. Поэтому кладоискатели хотят идти напрямую в точку, где зарыт клад. Например, вместо того, чтобы проходить три шага на север, один шаг на восток, один шаг на север, три шага на восток, два шага на юг и один шаг на запад, можно пройти напрямую, использовав около 3.6 шага (см. рисунок).




Вам необходимо узнать точку, в которой находится клад согласно указаниям пиратов.

**Входные данные**

Первая строка входного файла INPUT.TXT содержит число N – число указаний ($1 \le N \le 40$). Последующие N строк содержат сами указания – номер направления (целое число от 1 до 8) и количество шагов (целое число от 1 до 1000). Числа разделены пробелами.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите координаты X и Y точки (два вещественных числа, разделённые пробелом), где зарыт клад, считая, что ось OX направлена на восток, а ось OY – на север. В начале кладоискатель должен стоять в начале координат. Координаты необходимо вывести с округлением до тысячных с тремя знаками после запятой.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 6<br>1 3<br>3 1<br>1 1<br>3 3<br>5 2<br>7 1 | 3.000 2.000 |
| 2 | 1<br>8 10 | -7.071 7.071 |

## 208. Забавная игра

*(Время: 1 сек. Память: 16 Мб Сложность: 30%)*

Легендарный учитель математики Юрий Петрович придумал забавную игру с числами. А именно, взяв произвольное целое число, он переводит его в двоичную систему счисления, получая некоторую последовательность из нулей и единиц, начинающуюся с единицы. (Например, десятичное число $19_{10} = 1*2^4+0*2^3+0*2^2+1*2^1+1*2^0$ в двоичной системе запишется как $10011_2$.) Затем учитель начинает сдвигать цифры полученного двоичного числа по циклу (так, что последняя цифра становится первой, а все остальные сдвигаются на одну позицию вправо), выписывая образующиеся при этом последовательности из нулей и единиц в столбик – он подметил, что независимо от выбора исходного числа получающиеся последовательности начинают с некоторого момента повторяться. И, наконец, Юрий Петрович оты-

скивает максимальное из выписанных чисел и переводит его обратно в десятичную систему счисления, считая это число результатом проделанных манипуляций. Так, для числа 19 список последовательностей будет таким:

10011
11001
11100
01110
00111
10011
…

и результатом игры, следовательно, окажется число $1*2^4+1*2^3+1*2^2+0*2^1+0*2^0 = 28$.

Поскольку придуманная игра с числами все больше занимает воображение учителя, отвлекая тем самым его от работы с ну очень одаренными школьниками, Вас просят написать программу, которая бы помогла Юрию Петровичу получать результат игры без утомительных ручных вычислений.

**Входные данные**

Входной файл INPUT.TXT содержит одно целое число N ($0 \le N \le 32767$).

**Выходные данные**

Ваша программа должна вывести в выходной файл OUTPUT.TXT одно целое число, равное результату игры.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 19 | 28 |
| 2 | 1212 | 1938 |

## 209. Целые точки

*(Время: 1 сек. Память: 16 Мб Сложность: 64%)*

Многоугольник (не обязательно выпуклый) на плоскости задан координатами своих вершин. Требуется подсчитать количество точек с целочисленными координатами, лежащих внутри него (но не на его границе).

**Входные данные**

В первой строке входного файла INPUT.TXT содержится N ($3 \le N \le 10^3$) – число вершин многоугольника. В последующих N строках идут координаты ($X_i$, $Y_i$) вершин многоугольника в порядке обхода по часовой стрелке. $X_i$ и $Y_i$ - целые числа, по модулю не превосходящие $10^6$.

**Выходные данные**

Ваша программа должна вывести в выходной файл OUTPUT.TXT одно целое число - ответ на задачу.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 4<br>-1 -1<br>-1 1<br>1 1<br>1 -1 | 1 |
| 2 | 3<br>0 0<br>0 2<br>2 0 | 0 |

## 210. Степень

*(Время: 1 сек. Память: 16 Мб Сложность: 59%)*

Для того чтобы проверить, как её ученики умеют считать, Мария Ивановна каждый год задаёт им на дом одну и ту же задачу – для заданного натурального A найти минимальное натуральное N такое, что N в степени N (N, умноженное на себя N раз) делится на A. От года к году меняется только число A.

Вы решили помочь будущим поколениям. Для этого вам необходимо написать программу, решающую эту задачу.

**Входные данные**

Во входном файле INPUT.TXT содержится единственное число A ($1 \le A \le 10^9$ – на всякий случай, вдруг Мария Ивановна задаст большое число, чтобы «завалить» кого-нибудь…).

**Выходные данные**

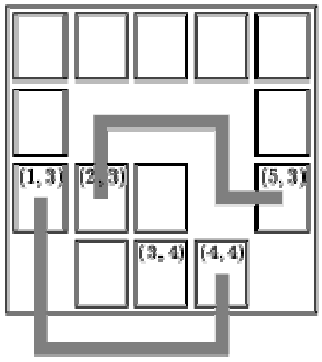
В выходной файл OUTPUT.TXT необходимо вывести единственное число N.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 8 | 4 |
| 2 | 13 | 13 |

## 211. Игра с фишками

*(Время: 1 сек. Память: 16 Мб Сложность: 52%)*




Вы являетесь одним из разработчиков новой компьютерной игры. Игра происходит на прямоугольной доске, состоящей из W×H клеток. Каждая клетка может либо содержать, либо не содержать фишку. Важной частью игры является проверка того, соединены ли две фишки путем, удовлетворяющим следующим свойствам:

Путь должен состоять из отрезков вертикальных и горизонтальных прямых.

Путь не должен пересекать других фишек. При этом часть пути может оказаться вне доски. Например:

Фишки с координатами (1, 3) и (4, 4) могут быть соединены. Фишки с координатами (2, 3) и (5, 3) тоже могут быть соединены. А вот фишки с координатами (2, 3) и (3, 4) соединить нельзя – любой соединяющий их путь пересекает другие фишки.

Вам необходимо написать программу, проверяющую, можно ли соединить две фишки путем, обладающим вышеуказанными свойствами, и, в случае положительного ответа, определяющую минимальную длину такого пути (считается, что путь имеет изломы, начало и конец только в центрах клеток (или «мнимых клеток», расположенных вне доски), а отрезок, соединяющий центры двух соседних клеток, имеет длину 1).

**Входные данные**

Первая строка входного файла INPUT.TXT содержит два натуральных числа: W – ширина доски, H – высота доски ($1 \le W, H \le 75$). Следующие H строк содержат описание доски: каждая строка состоит ровно из W символов: символ «X» (заглавная английская буква «экс») обозначает фишку, символ «.» (точка) обозначает пустое место. Все остальные строки содержат описания запросов: каждый запрос состоит из четырёх натуральных чисел, разделённых пробелами – X1, Y1, X2, Y2, причём $1 \le X1, X2 \le W$, $1 \le Y1, Y2 \le H$. Здесь (X1, Y1) и (X2, Y2) – координаты фишек, которые требуется соединить (левая верхняя клетка имеет координаты (1, 1)). Гарантируется, что эти координаты не будут совпадать (кроме последнего запроса; см. далее). Последняя строка содержит запрос, состоящий из четырёх чисел 0; этот запрос обрабатывать не надо. Количество запросов не превосходит 20.

**Выходные данные**

Для каждого запроса в выходной файл OUTPUT.TXT необходимо вывести одно целое число на отдельной строке – длину кратчайшего пути, или 0, если такого пути не существует.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 5 4<br>XXXXX<br>X...X<br>XXX.X<br>.XXX.<br>2 3 5 3<br>1 3 4 4<br>2 3 3 4<br>0 0 0 0 | 5<br>6<br>0 |

## 212. Деревни

*(Время: 1 сек. Память: 16 Мб Сложность: 79%)*

В тридесятом государстве есть N деревень. Некоторые пары деревень соединены дорогами. В целях экономии, «лишних» дорог нет, т.е. из любой деревни в любую можно добраться по дорогам единственным образом.

Новейшие исследования показали, что тридесятое государство находится в сейсмически опасной зоне. Поэтому глава государства захотел узнать, какой именно ущерб может принести его державе землетрясение. А именно, он хочет узнать, какое минимальное число дорог должно быть разрушено, чтобы образовалась изолированная от остальных группа ровно из P деревень такая, что из любой деревни из этой группы до любой другой деревни из этой группы по-прежнему можно будет добраться по неразрушенным дорогам (группа изолирована от остальных, если никакая неразрушенная дорога не соединяет деревню из этой группы с деревней не из этой группы).

Вы должны написать программу, помогающую ему в этом.

**Входные данные**

Первая строка входного файла INPUT.TXT содержит два числа: N и P ($1 \le P \le N \le 150$). Все остальные строки содержат описания дорог, по одному на строке: описание дороги состоит из двух номеров деревень (от 1 до N), которые эта дорога соединяет. Все числа во входном файле разделены пробелами и/или переводами строки.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите единственное число – искомое количество дорог.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3 2<br>1 2<br>3 2 | 1 |
| 2 | 11 6<br>1 2<br>1 3<br>1 4<br>1 5<br>2 6<br>2 7<br>2 8<br>4 9<br>4 10<br>4 11 | 2 |

## 213. Подсчет баллов

*(Время: 1 сек. Память: 16 Мб Сложность: 42%)*

Решение каждой задачи заочного тура проверяется на наборе заранее заготовленных тестов. По результатам работы программы на каждом тесте участнику либо начисляются баллы за этот тест (когда программа выдала правильный ответ), либо не начисляются (когда во время работы программы произошли ошибки или выданный ответ не верен). Тесты могут иметь разную стоимость.

Дополнительные баллы начисляются участнику, если его программа прошла все тесты.

Участник может исправлять свое решение, и посылать его на проверку повторно (при этом решение проверяется на том же наборе тестов). При этом за каждую попытку из количества набранных по задаче баллов вычитается штраф, который равен 0 при 1-й попытке, а при каждой следующей возрастает на 2 (то есть 2 при второй, 4 – при третьей, 6 – при четвертой и т.д.).

Из баллов, полученных участником за каждую из попыток (с учетом начисленных штрафов), выбирается максимальный результат, который и засчитывается как результат данного участника по этой задаче. Это нужно, в частности, для того, чтобы последующие попытки не ухудшали уже полученный участником результат по задаче.

Например, если участник делает первую попытку и набирает 10 баллов, его результат по задаче равен 10 баллов. Пусть на второй попытке участник посылает решение, которое набирает 8 баллов. С учетом штрафа за эту попытку участник имеет 6 баллов, однако результат команды по задаче остается равным 10. Пусть с 3-й попытки решение набрало 20 баллов, тогда (с учетом штрафа) результат участника по задаче становится равен 16 баллам. Наконец, пусть с 4-й попытки решение проходит все тесты, тогда участник получает сумму баллов за все тесты, плюс призовые баллы за прохождение всех тестов, минус 6 баллов штрафа (если, конечно, эта величина не меньше 16 баллов, которые уже были у данного участника).

Напишите программу, которая определяет результат данного участника по этой задаче.

**Входные данные**

Во входном файле INPUT.TXT записано сначала число N – количество тестов, на которых проверяются решения данной задачи ($1 \le N \le 100$). Далее идет N натуральных чисел, не превышающих 100, – баллы, которые начисляются за прохождение каждого из тестов. Далее идет целое число из диапазона от 0 до 100 – количество баллов, которое дополнительно начисляется за прохождение всех тестов.

Далее идет натуральное число M – количество попыток сдачи задачи ($1 \le M \le 100$). После чего идет M наборов по N чисел в каждом, задающих результаты проверки каждой из M попыток сдачи задачи на тестах. 0 обозначает, что соответствующий тест не пройден, 1 – пройден.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите M чисел. i-ое число должно соответствовать результату участника после совершения им первых i попыток.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 4<br>1 2 3 4<br>5<br>3<br>0 0 0 0<br>1 1 1 1<br>0 1 0 1 | 0<br>13<br>13 |

## 214. Великая сеча

*(Время: 1 сек. Память: 16 Мб Сложность: 58%)*

Алеша Попович и Добрыня Никитич сражаются со стаей двух- и трехголовых драконов. Они по очереди взмахивают мечами, и одним махом могут отрубить любое (по своему желанию) число голов, но только у одного дракона. Отрубивший последнюю голову у последнего дракона получает в жены прекрасную принцессу.

Кто из богатырей (начинающий или второй) может получить в жены принцессу независимо от действий другого?

**Входные данные**

Во входном файле INPUT.TXT записано два числа N и M – количество двух- и трехголовых драконов соответственно (оба числа целые из диапазона от 0 до 100, N+M>0).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите число 1 или 2 определяющее, кто из богатырей имеет все шансы получить в жены принцессу (1 – тот, кто начинает, 2 – второй).

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 2 2 | 2 |
| 2 | 1 2 | 1 |

## 215. Водостоки

*(Время: 1 сек. Память: 16 Мб Сложность: 55%)*

Карту местности условно разбили на квадраты, и посчитали среднюю высоту над уровнем моря для каждого квадрата.

Когда идет дождь, вода равномерно выпадает на все квадраты. Если один из четырех соседних с данным квадратом квадратов имеет меньшую высоту над уровнем моря, то вода с текущего квадрата стекает туда (и, если есть возможность, то дальше), если же все соседние квадраты имеют большую высоту, то вода скапливается в этом квадрате.

Разрешается в некоторых квадратах построить водостоки. Когда на каком-то квадрате строят водосток, то вся вода, которая раньше скапливалась в этом квадрате, будет утекать в водосток.

Если есть группа квадратов, имеющих одинаковую высоту и образующих связную область, то если хотя бы рядом с одним из этих квадратов есть квадрат, имеющий меньшую высоту, то вся вода утекает туда, если же такого квадрата нет, то вода стоит во всех этих квадратах. При этом достаточно построить водосток в любом из этих квадратов, и вся вода с них будет утекать в этот водосток.

Требуется определить, какое минимальное количество водостоков нужно построить, чтобы после дождя вся вода утекала в водостоки.

**Входные данные**

Во входном файле INPUT.TXT записаны сначала числа N и M, задающие размеры карты – натуральные числа, не превышающие 100. Далее, идет N строк, по M чисел в каждой, задающих высоту квадратов карты над уровнем моря. Высота задается натуральным числом, не превышающим 10000. Считается, что квадраты, расположенные за пределами карты, имеют высоту 10001 (то есть вода никогда не утекает за пределы карты).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите минимальное количество водостоков, которое необходимо построить.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 4 5<br>1 2 3 1 10<br>1 4 3 10 10<br>1 5 5 5 5<br>6 6 6 6 6 | 2 |

## 216. Коллекционирование этикеток

*(Время: 1 сек. Память: 16 Мб Сложность: 61%)*

Вася коллекционирует спичечные этикетки. Для этого у него есть N альбомов вместимостью $K_1$, $K_2$, …, $K_N$ этикеток. Вася хочет, чтобы в случае утери одного любого альбома каждая этикетка осталась у него хотя бы в одном экземпляре. Для этого он

покупает каждую этикетку в двух экземплярах, и наклеивает их в два разных альбома. Какое максимальное количество различных этикеток при этом может оказаться в его коллекции?

**Входные данные**

Входной файл INPUT.TXT содержит сначала число N – количество альбомов, а затем N чисел $K_1$, $K_2$, …, $K_N$, задающих вместимости альбомов. N – натуральное число из диапазона от 2 до 1000. Вместимость каждого альбома задается натуральным числом, суммарная вместимость всех альбомов не превышает 100 000 этикеток.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите число E – максимальное количество различных этикеток, которое может собрать Вася с соблюдением выдвинутого условия.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 4<br>1 2 1 1 | 2 |

## 217. Еловая аллея

*(Время: 1 сек. Память: 16 Мб Сложность: 47%)*

Мэр города Урюпинска решил посадить на главной аллее города, которая проходит с запада на восток, голубые ели. Причем сажать ели можно не во всех местах, а только на специально оставленных при асфальтировании аллеи клумбах.

Как оказалось, голубые ели бывают M различных сортов. Для ели каждого сорта известна максимальная длина ее тени в течение дня в западном и в восточном направлении ($W_i$ и $E_i$, соответственно). При этом известно, что ели растут гораздо лучше, если в течение дня они не оказываются в тени других елей.

Координатная ось направлена вдоль аллеи с запада на восток.

По заданным координатам клумб вычислите максимальное число елей, которое можно посадить, соблюдая условие о том, что никакая ель не должна попадать в тень от другой ели.

**Входные данные**

Во входном файле INPUT.TXT записано сначала натуральное число M – количество сортов елей ($1 \le M \le 100$). Затем идет M пар чисел $W_i$, $E_i$, описывающих максимальную длину тени в западном и восточном направлении в течение дня для каждого сорта ели (числа $W_i$, $E_i$ – целые, из диапазона от 0 до 30000). Далее идет натуральное число N – количество клумб, в которых можно сажать ели ($1 \le N \le 100$). Далее идет N чисел, задающих координаты клумб (координаты – целые числа, по модулю не превышающие 30000). Клумбы перечислены с запада на восток (в порядке возрастания их координат).

**Примечание**

Если на клумбе с координатой X мы посадили ель, максимальная тень которой в восточном направлении равна E, то все клумбы с координатами от X+1 до X+E–1 попадают в тень от этой ели, а клумба с координатами X+E – уже нет. Аналогично для тени в западном направлении.

**Выходные данные**

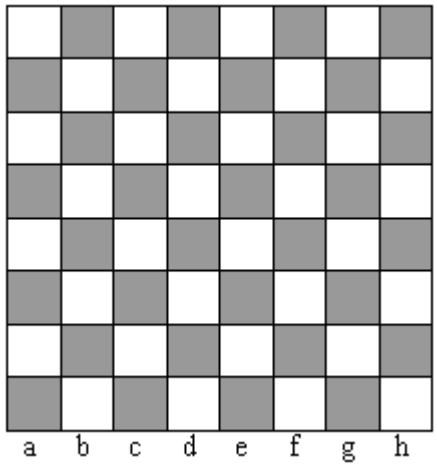
В выходной файл OUTPUT.TXT выведите число A – максимальное количество елей, которые удастся посадить.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3<br>1000 3<br>1 200<br>128 256<br>3<br>1 2 4 | 2 |

# 218. Шашки

*(Время: 1 сек. Память: 16 Мб Сложность: 45%)*




Петя и Вася играли в шашки по описанным ниже правилам. В какой-то момент забежавший в комнату кот перевернул доску, на которой играли Петя и Вася. К счастью, у них осталась запись сделанных ходов, используя которую можно восстановить расположение шашек к моменту, когда забежал кот.

Напишите программу, которая выведет положение шашек на доске после выполнения описанных ходов.

Игра происходит на стандартной доске (8x8), которая располагается так, что у игрока, играющего белыми, левая нижняя клетка является черной, и с нее начинается нумерация как строк, так и столбцов. Строки доски нумеруются числами от 1 до 8, столбцы – латинскими буквами от a до h.

В начале игры каждый из двух игроков имеет по 12 шашек своего цвета (белого или черного соответственно). Белые шашки располагаются на клетках a1, a3, b2, c1, c3, d2, e1, e3, f2, g1, g3, h2. Черные шашки располагаются на клетках a7, b6, b8, c7, d6, d8, e7, f6, f8, g7, h6, h8.

Игроки совершают ходы по очереди. Игрок, играющий белыми, ходит первым.

Шашки в процессе игры бывают двух видов: обычная шашка и дамка. В начале игры все шашки обычные. Белая шашка становится дамкой, если она оказывается в строке 8. Соответственно, черная шашка становится дамкой, если она оказывается в строке 1.

*Шашка* может совершать ходы двух типов:

1. *Простой ход* заключается в перемещении одной из шашек на одну клетку вперед по диагонали. Например, белая шашка с e3 может сходить на d4 или f4 (если соответствующая клетка свободна). А черная шашка с e3 может сходить на d2 или f2.

2. *Рубка* заключается в том, что шашка перепрыгивает через шашку (или дамку) противника, находящуюся в диагонально соседней с ней клетке при условии, что следующая клетка этой диагонали свободна. Шашка противника, которую срубили, убирается с доски. Если сразу после окончания рубки та же самая шашка может продолжить рубку, она ее продолжает этим же ходом. Рубка возможна в любом из 4-х диагональных направлений. Если в процессе рубки шашка оказывается в 1-й строке (для черных) или в 8-й (для белых), она становится дамкой.

*Дамка* может совершать следующие ходы:

3. *Простой ход* заключается в перемещении дамки по диагонали на любое число клеток (при этом все клетки, через которые происходит перемещение, должны быть свободны).

4. *Рубка* заключается в том, что шашка перепрыгивает через шашку (или дамку) противника, находящуюся на той же диагонали, что и рубящая дамка. Это можно делать при условии, что все клетки между рубящей дамкой и шашкой, которую рубят, а также клетка, следующая за шашкой, которую рубят, свободны. После рубки дамка может встать на любую клетку данной диагонали за клеткой, на которой стояла срубленная шашка (при условии, что все клетки на ее пути свободны). Если из своего нового положения дамка может совершить рубку, она должна ее совершить этим же ходом.

**Входные данные**

В первой строке входного файла INPUT.TXT записано одно число N – количество ходов, которое было сделано с начала партии. Это количество не превышает 1000.

В каждой из следующих N строк записаны описания ходов (нечетные ходы совершались белыми, четные – черными). Описание каждого хода занимает ровно одну строку и записывается в следующем виде.

Простой ход записывается в виде <начальная клетка>–<конечная клетка>. Например, ход с c3 на d4 записывается как c3-d4.

Рубка записывается в виде <начальная клетка>:<клетка после рубки>. Если рубка продолжается, то ставится еще одно двоеточие, и записывается клетка, где окажется шашка после следующей рубки и т.д. Например, e3:c5:e7 (шашка, изначально расположенная на e3, рубит шашку на d4 и оказывается на c5, после чего рубит шашку на d6 и оказывается на e7).

Рубка a1:h8 может быть осуществлена только дамкой (например, дамка с a1 рубит шашку, стоящую в c3, и заканчивает ход в h8). Рубка дамкой может быть и такой a1:d4:f6:h4 (дамка рубит шашку, стоящую на b2, затем продолжает рубку и рубит шашку на e5, и, наконец, рубит шашку на g5). При этом после каждой рубки указывается клетка, на которой останавливается дамка перед следующей рубкой.

Строки с описанием ходов не содержат пробельных символов.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите изображение доски с расположенными на ней шашками. В первой строке выходного файла должна быть выведена 8-я строка доски, во второй – 7-я и т.д. В каждой строке должно быть ровно 8 символов, описывающих клетки столбцов с a по h.

Белая клетка должна быть изображена символом «.» (точка), пустая черная клетка – символом «–» (минус). Черная клетка, в которой стоит белая шашка – символом «w» (маленькая латинская буква w), а клетка с белой дамкой – символом «W» (заглавная латинская буква W). Аналогично клетка с черной шашкой должна быть изображена символом «b» (маленькая латинская буква b), а клетка с черной дамкой – символом «B» (большая латинская буква B).

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3<br>c3-d4<br>f6-e5<br>d4:f6 | .b.b.b.b<br>b.b.b.b.<br>.b.b.w.b<br>-.-.-.-.<br>.-.-.-.-<br>w.-.w.w.<br>.w.w.w.w<br>w.w.w.w. |

## 219. Симпатичные таблицы

*(Время: 1 сек. Память: 16 Мб Сложность: 75%)*

Рассмотрим таблицу размера MxN, в клетках которой стоят целые неотрицательные числа. Скажем, что таблица является симпатичной, если для всех i сумма чисел ее i-ой строки не превышает $R_i$ и для всех j сумма чисел ее j-го столбца не превышает $C_j$.

Вам задана таблица Z размера MxN (N строк и M столбцов), в некоторых клетках которой уже стоят целые неотрицательные числа. Найдите симпатичную таблицу с максимальной суммой элементов такую, что она совпадает с Z на тех клетках, в которых в Z стоят числа.

**Входные данные**

Первая строка входного файла INPUT.TXT содержит числа N и M ($1 \le M, N \le 20$). Следующая строка содержит N целых неотрицательных чисел - $R_1$, $R_2$, ..., $R_N$. Следующая строка содержит M целых неотрицательных чисел $C_1$, $C_2$, ..., $C_M$. Все числа не превышают $10^6$. Следующие N строк содержат по M целых чисел, которые задают Z. Если на некотором месте в таблице отсутствует число, то на этом месте во входном файле стоит число -1.

**Выходные данные**

Выведите в выходной файл OUTPUT.TXT выведите целое число – сумму элементов найденной таблицы. Если решение не существует, то выведите единственное число -1.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 2 2<br>2 2<br>1 2<br>1 -1<br>-1 -1 | 3 |

## 220. Мышка с колесиком

*(Время: 1 сек. Память: 16 Мб Сложность: 54%)*

Пользователь просматривает таблицу в Internet Explorer и пользуется для прокрутки изображения колесиком на мышке. При этом все изображение сдвигается вверх или вниз на T пикселей. Пользователю очень не нравится, когда курсор мыши оказывается на горизонтальных линиях, разделяющих строки таблицы. Поэтому он хочет выбрать такое положение для курсора мыши на экране, чтобы в процессе прокрутки до конца таблицы курсор как можно меньшее число раз пересекался с линиями таблицы.

При этом если в каком-то положении курсор оказывается на двух линиях таблицы, то это считается за два пересечения курсора с линиями таблицы. Если какую-то линию курсор мыши пересекает в двух положениях (то есть, например, высота курсора 10 пикселей, а при прокрутке таблица сдвигается на 7 пикселей, тогда курсор мыши может оказываться на одной линии в двух состояниях прокрутки), то это также считается за два пересечения.

Экран монитора имеет разрешение по вертикали U пикселей. Координаты введены так, что самые верхние точки экрана имеют координату 0, а нижние – координату U–1. Курсор мыши имеет высоту H пикселей. Расположением курсора считается самая верхняя точка курсора. Таким образом, если мы говорим, что он расположен, например, в точке с координатами 0 на экране, то его изображение расположено в точках с координатами от 0 до H–1. Курсор мыши всегда целиком помещается на экране, то есть допустимыми координатами для его расположения являются координаты от 0 до U–H. Таблица, которую просматривает пользователь, имеет высоту L пикселей и состоит из N–1 строки, и, следовательно, в ней N горизонтальных линий, которые имеют координаты $X_1, X_2, \ldots, X_N$. При этом $0 = X_1 < X_2 < X_3 < \ldots < X_N = L–1$.

В начальный момент времени таблица расположена так, что линия, имеющая координату 0 в таблице отображается в 0-й строке пикселей монитора. Далее при прокрутке таблица каждый раз сдвигается на T пикселей (то есть в 0-й строке монитора оказывается строка пикселей, имеющая в таблице координату T, координату 2T и т.д.). Так происходит до тех пор, пока на экране не окажется нижняя линия таблицы (которая имеет координату $X_N$). После этого дальнейшая прокрутка не происходит (если изначально $X_N < U$, то прокрутка вообще не происходит).

**Входные данные**

Во входном файле INPUT.TXT задано сначала разрешение монитора по вертикали U, затем высота курсора мыши H, затем шаг прокрутки T. Далее задана высота таблицы L. Далее задано количество разделительных линий в таблице N, и координаты X1, X2,…,XN, где расположены эти линии относительно начала таблицы. Ограничения: $10 \le U \le 512$, $1 \le H \le U$, $1 \le T \le U$, $2 \le N \le 200000$, $0 = X_1 < X_2 < X_3 < \ldots < X_N = L–1 \le 10^9$.

**Выходные данные**

В выходной файл OUTPUT.TXT нужно вывести наименьшее возможное количество пересечений курсора мыши с линиями таблицы.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 10 3 10 10<br>4<br>0 1 6 9 | 0 |
| 2 | 10 6 2 21<br>11<br>0 2 4 6 8 10 12 14 16 18 20 | 21 |

## 221. Левый лабиринт

*(Время: 1 сек. Память: 16 Мб Сложность: 49%)*

В спортзале размером NxM метров построили современный аттракцион под названием «Левый лабиринт». Для этого на полу спортзала с интервалом в 1 метр начертили линии, параллельные стенам спортзала. Таким образом, спортзал разбили на NxM клеток. Дальше некоторые из этих клеток покрасили в черный цвет. Аттракцион заключается в том, что участника ставят в некоторой клетке спортзала и просят как можно быстрее добежать до некоторой другой клетки. При этом накладываются следующие условия:

- Участнику запрещено ходить по черным клеткам.
- Придя в какую-то клетку, участник может пойти либо прямо, либо налево, либо направо (если в соответствующем направлении клетка не покрашена в черный цвет): ходить назад, а также ходить по диагонали запрещается.
- За все время пути участнику разрешается повернуть направо (то есть пойти из текущей клетки направо относительно того, откуда он пришел в данную клетку) не более K раз.
- В начальной клетке участник может встать лицом в ту сторону, в какую ему захочется. С какой стороны участник прибежит в конечную клетку также не важно.

Известно, что на то, чтобы перебежать из клетки в соседнюю, участник тратит ровно 1 секунду. Требуется вычислить минимальное время, за которое участник сможет достичь конечной клетки.

**Входные данные**

Во входном файле INPUT.TXT сначала записано число K – количество разрешенных поворотов направо (целое число из диапазона от 0 до 5), затем записаны числа N и M, задающие размеры спортзала – натуральные числа, не превышающие 20. Далее записано N строк по M чисел в каждой. Число 0 обозначает неокрашенную клетку, число 1 – покрашенную, число 2 – клетку, откуда стартует участник и число 3 – клетку, куда нужно добежать (клетки, помеченные 2 и 3 являются неокрашенными). В лабиринте всегда есть ровно одна начальная клетка и ровно одна клетка, в которую нужно попасть.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите минимальное время, за которое можно добраться в конечную клетку. Если попасть в конечную клетку нельзя, выведите –1.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 1 5 5<br>2 0 0 1 1<br>0 1 0 0 1<br>1 1 0 0 1<br>0 0 0 0 1<br>3 1 0 0 1 | 12 |
| 2 | 0 1 3<br>2 1 3 | -1 |

## 222. Дремучий лес

*(Время: 1 сек. Память: 16 Мб Сложность: 50%)*

Будем говорить, что для наблюдателя лес является дремучим, если из своего текущего положения наблюдатель видит только деревья. Вам дана карта леса и координаты точки, в которой находится наблюдатель. Требуется определить, кажется ли лес дремучим данному наблюдателю.

На карте леса все деревья изображаются кругами. При этом в лесу бывают сросшиеся деревья (изображения таких деревьев на карте пересекаются), также одно дерево может находиться внутри другого. Точка, в которой стоит наблюдатель, не лежит внутри или на границе ни одного из деревьев.

**Входные данные**

Во входном файле INPUT.TXT содержится сначала целое число N – количество деревьев ($1 \le N \le 50000$). Затем идут два числа, задающих координаты наблюдателя. Затем идет N троек чисел, задающих деревья. Первые два числа задают координаты центра, а третье – радиус. Все координаты задаются точно, и выражаются вещественными числами не более чем с 2 знаками после десятичной точки, по модулю не превосходящими 100000.

**Выходные данные**

В выходной файл OUTPUT.TXT нужно вывести сообщение YES, если лес является дремучим, и NO иначе.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 4<br>0 0<br>2 2 2<br>-2 2 2<br>-2 -2 2<br>2 -2 2 | YES |
| 2 | 2<br>10 10<br>0 0 1<br>0.5 0 2 | NO |

## 223. Анаграммер

*(Время: 1 сек. Память: 16 Мб Сложность: 65%)*

Анаграммер – специальное устройство для получения из слова его анаграмм (то есть слов, записанных теми же буквами, но в другом порядке). Это устройство умеет выполнять 2 операции:

1. Взять очередную букву исходного слова и поместить ее в стек.
2. Взять букву из стека и добавить ее в конец выходного слова.

Стек – это хранилище данных, которое работает по принципу «первый пришел – последний ушел». Стек можно представить себе в виде пирамидки. Когда мы добавляем букву в стек, это соответствует тому, что на стержень пирамидки сверху мы надеваем кольцо, на котором написана соответствующая буква. Когда берем букву из стека, то это соответствует тому, что мы снимаем со стержня верхнее кольцо, и смотрим, какая буква на нем написана.

Например, слово TROT в слово TORT может быть преобразовано анаграммером двумя различными последовательностями операций: 11112222 или 12112212.

Напишите программу, которая по двум заданным словам вычисляет количество различных последовательностей операций анаграммера, которые преобразуют первое из этих слов во второе.

**Входные данные**

Первая строка входного файла INPUT.TXT содержит исходное слово, а вторая – слово, которое необходимо получить. Слова состоят только из заглавных латинских букв и имеют длину не более 50 символов. Оба слова имеют одинаковую длину. В этих строках не содержится пробелов.

**Выходные данные**

В первой строке выходного файла OUTPUT.TXT должно содержаться количество последовательностей операций анаграммера, с помощью которых можно преобразовать первое слово во второе.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | TROT<br>TORT | 2 |
| 2 | MADAM<br>ADAMM | 4 |
| 3 | LONG<br>GONG | 0 |
| 4 | AAAAAAAA<br>AAAAAAAA | 1430 |

## 224. Наибольшее произведение

*(Время: 1 сек. Память: 16 Мб Сложность: 36%)*

Дано N целых чисел. Требуется выбрать из них три таких числа, произведение которых максимально.

**Входные данные**

Во входном файле INPUT.TXT записано сначала число N – количество чисел в последовательности ($3 \le N \le 10^6$). Далее записана сама последовательность: N целых чисел, по модулю не превышающих 30000.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите значение наибольшего произведения искомых трех чисел.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 9<br>3 5 1 7 9 0 9 -3 10 | 810 |
| 2 | 3<br>-5 -30000 -12 | -1800000 |

## 225. Покупка билетов

*(Время: 1 сек. Память: 16 Мб Сложность: 43%)*

За билетами на премьеру нового мюзикла выстроилась очередь из N человек, каждый из которых хочет купить 1 билет. На всю очередь работала только одна касса, поэтому продажа билетов шла очень медленно, приводя «постояльцев» очереди в отчаяние. Самые сообразительные быстро заметили, что, как правило, несколько билетов в одни руки кассир продаёт быстрее, чем когда эти же билеты продаются по одному. Поэтому они предложили нескольким подряд стоящим людям отдавать деньги первому из них, чтобы он купил билеты на всех.

Однако для борьбы со спекулянтами кассир продавала не более 3-х билетов в одни руки, поэтому договориться таким образом между собой могли лишь 2 или 3 подряд стоящих человека.

Известно, что на продажу i-му человеку из очереди одного билета кассир тратит Ai секунд, на продажу двух билетов – Bi секунд, трех билетов – Ci секунд. Напишите программу, которая подсчитает минимальное время, за которое могли быть обслужены все покупатели.

Обратите внимание, что билеты на группу объединившихся людей всегда покупает первый из них. Также никто в целях ускорения не покупает лишних билетов (то есть билетов, которые никому не нужны).

**Входные данные**

Во входном файле INPUT.TXT записано сначала число N – количество покупателей в очереди ($1 \le N \le 5000$). Далее идет N троек натуральных чисел Ai, Bi, Ci. Каждое из этих чисел не превышает 3600. Люди в очереди нумеруются, начиная от кассы.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите одно число – минимальное время в секундах, за которое могли быть обслужены все покупатели.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 5<br>5 10 15<br>2 10 15<br>5 5 5<br>20 20 1<br>20 1 1 | 12 |
| 2 | 2<br>3 4 5<br>1 1 1 | 4 |

## 226. Перегоны

*(Время: 1 сек. Память: 16 Мб Сложность: 50%)*

На некоторой железнодорожной ветке расположено N станций, которые последовательно пронумерованы числами от 1 до N. Известны расстояния между некоторыми станциями. Требуется точно вычислить длины всех перегонов между соседними станциями или указать, что это сделать невозможно (то есть приведенная информация является противоречивой или ее недостаточно).

**Входные данные**

Во входном файле INPUT.TXT записаны сначала числа N – количество станций ($2 \le N \le 100$) и E – количество пар станций, расстояния между которыми заданы ($0 \le E \le 10000$). Далее, идет E троек чисел, первые два числа каждой тройки задают номера станций (это числа из диапазона от 1 до N), а третье – расстояние между этими станциями (все эти расстояния заданы точно и выражаются вещественными неотрицательными числами не более чем с 3-я знаками после десятичной точки).

**Выходные данные**

В случае, когда восстановить длины перегонов можно однозначно, в выходной файл OUTPUT.TXT выведите сначала «YES», а затем N–1 вещественное число. Первое из этих чисел должно соответствовать расстоянию от 1-й станции до 2-й, второе – от 2-й до 3-й, и так далее. Все числа должны быть выведены с точностью до 3-х знаков после десятичной точки (например, число 2.3 следует выводить как 2.300). Если приведенная информация о расстояниях между станциями является противоречивой или не позволяет однозначно точно восстановить длины перегонов, выведите в выходной файл «NO».

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3 2<br>1 2 1.250<br>3 1 3 | YES<br>1.250 1.750 |
| 2 | 4 4<br>1 2 1.250<br>3 1 1.255<br>2 4 0.010<br>1 1 0.000 | YES<br>1.250 0.005 0.005 |
| 3 | 3 1<br>1 1 1 | NO |
| 4 | 3 3<br>1 2 1.250<br>1 3 1.300<br>2 3 1.000 | NO |

## 227. Сломанный калькулятор

*(Время: 1 сек. Память: 16 Мб Сложность: 62%)*

У калькулятора есть две ячейки памяти: содержимое первой из них всегда отображается на табло, вторая является буфером. В начальный момент времени на табло калькулятора отображается целое число X, а в буфере записано число 0. У калькулятора работают только две клавиши: «+» и «=». При нажатии на «+» число, которое в данный момент отображено на табло, копируется в буфер. При нажатии на «=» число из буфера прибавляется к числу, отображенному на табло, и результат отображается на табло, число в буфере при этом не меняется.

Требуется за наименьшее число нажатий клавиш на калькуляторе добиться того, чтобы на табло было отображено число Y.

**Входные данные**

Входной файл INPUT.TXT содержит два целых числа X и Y. Каждое из этих чисел по модулю не превышает $10^9$.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите одно число – количество нажатий клавиш, которое потребуется для получения числа Y. Если из числа X получить число Y с помощью указанных операций невозможно, в выходной файл выведите одно число –1.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT | Пояснение |
|---|---|---|---|
| 1 | 1 1 | 0 | |
| 2 | -2 -6 | 3 | +== |
| 3 | 1 8 | 6 | +===+= |
| 4 | 2 5 | -1 | |

## 228. Валютные махинации

*(Время: 1 сек. Память: 16 Мб Сложность: 39%)*

Петя, изучая, как меняется курс рубля по отношению к доллару и евро, вывел закон, по которому происходят эти изменения (или думает, что вывел :) ). По этому закону Петя рассчитал, каков будет курс рубля по отношению к доллару и евро в ближайшие N дней.

У Пети есть 100 рублей. В каждый из дней он может обменивать валюты друг на друга по текущему курсу без ограничения количества (при этом курс доллара по отношению к евро соответствует величине, которую можно получить, обменяв доллар на рубли, а потом эти рубли – на евро). Поскольку Петя будет оперировать не с наличной валютой, а со счетом в банке, то он может совершать операции обмена с любым (в том числе и нецелым) количеством единиц любой валюты.

Напишите программу, которая вычисляет, какое наибольшее количество рублей сможет получить Петя к исходу N-го дня.

Законы изменения курсов устроены так, что в течение указанного периода рублевый эквивалент той суммы, которая может оказаться у Пети, не превысит $10^8$ рублей.

**Входные данные**

Первая строка входного файла INPUT.TXT содержит одно число N ($1 \le N \le 5000$). В каждой из следующих N строк записано по 2 числа, вычисленных по Петиным законам для соответствующего дня – сколько рублей будет стоить 1 доллар, и сколько рублей будет стоить 1 евро. Все эти значения не меньше 0.01 и не больше 10000. Значения заданы точно и выражаются вещественными числами не более, чем с двумя знаками после десятичной точки.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите искомую величину с двумя знаками после десятичной точки.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 4<br>1 10<br>10 5.53<br>5.53 1.25<br>6 5 | 4000.00 |

## 229. Двухтуровая олимпиада

*(Время: 1 сек. Память: 16 Мб Сложность: 46%)*

Как известно, личная олимпиада по информатике проходит в два тура. На каждом из туров участники получают какие-то баллы, при этом итоговый балл определяется как сумма полученных баллов. Известны баллы, которые каждый участник получил на каждом из туров. Жюри хочет фальсифицировать итоги олимпиады так, чтобы победил «нужный» участник.

При этом жюри может делать следующие «подтасовки» (можно делать несколько «подтасовок» применительно как к одному и тому же, так и к разным турам):

1. Прибавить к результатам всех участников по одному из туров одно и то же положительное число.
2. Умножить результаты участников по одному из туров на некоторый коэффициент, больший 1.

При этом должна сохраниться правдоподобность результатов, которая заключается в том, что никто из участников не должен получить больше 100 баллов за каждый из туров.

Определите список участников, которые в результате таких фальсификаций могут оказаться победителями олимпиады (то есть в сумме за два тура иметь не меньше баллов, чем каждый из остальных участников).

**Входные данные**

Во входном файле INPUT.TXT записано сначала число участников N ($1 \le N \le 1000$), затем N пар чисел – результаты каждого участника за 1-й и за 2-й туры (результат участника за тур – это вещественное число от 0 до 100) не более, чем с 3 знаками после десятичной точки.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите сначала количество участников, которые смогут стать победителями олимпиады, а затем в возрастающем порядке их номера.

**Пример**

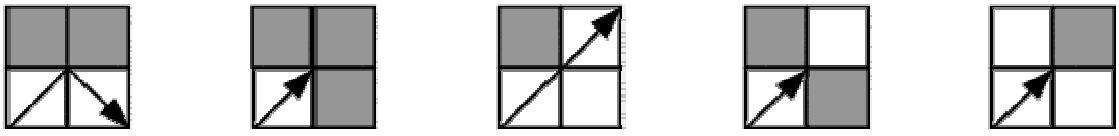
| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 4<br>45 90<br>70 80<br>0 0<br>75 75 | 2<br>2 4 |

## 230. Луч света в темном царстве

*(Время: 1 сек. Память: 16 Мб Сложность: 63%)*

Темное царство представляет собой лабиринт NxM, некоторые клетки которого окружены зеркальными стенами, а остальные – пустые. Весь лабиринт также окружен зеркальной стеной. В одной из пустых клеток лабиринта поставили светофор, который испускает лучи в 4 направлениях: под 45 градусов относительно стен лабиринта. Требуется изобразить траекторию этих лучей.

Когда луч приходит в угол, через который проходят зеркальные стены, дальше он идет так, как показано на рисунках (серым цветом показаны клетки, которые окружены зеркальными стенами). Аналогичным образом луч ведет себя, когда приходит на границу лабиринта.



**Входные данные**

В первой строке входного файла INPUT.TXT записаны два натуральных числа N и M – число строк и столбцов в лабиринте (каждое из чисел не меньше 1 и не больше 100). В следующих N строках записано ровно по M символов в каждой – карта лабиринта. Символ * (звездочка) обозначает клетку, окруженную зеркальными стенками, . (точка) – пустую клетку, символ X (заглавная латинская буква X) – клетку, в которой расположен светофор (такая клетка ровно одна).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите N строк по M символов в каждой – изображение лабиринта с траекториями лучей. Здесь, как и раньше, * (звездочка) должна обозначать клетки, окруженные зеркальными стенами, . (точка) – пустые клетки, через которые лучи света не проходят, / (слеш) – клетки, через которые луч света проходит из левого нижнего угла в правый верхний (или обратно – из правого верхнего в левый нижний), \ (обратный слеш) – клетки, через которые луч проходит из левого верхнего угла в правый нижний (или обратно), а символ X (заглавная латинская буква X) – клетки, через которые лучи проходят по обеим диагоналям.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 5 6<br>..*...<br>......<br>.....*<br>*X...*<br>....*. | ./*./\<br>/..X./<br>\./.X*<br>*X.//*<br>/\X/*. |
| 2 | 3 3<br>...<br>.X.<br>... | \./<br>.X.<br>/.\ |

## 231. Распаковка строки

*(Время: 1 сек. Память: 16 Мб Сложность: 25%)*

Будем рассматривать только строчки, состоящие из заглавных латинских букв. Например, рассмотрим строку AAAABCCCCCDDDD. Длина этой строки равна 14. Поскольку строка состоит только из латинских букв, повторяющиеся символы могут быть удалены и заменены числами, определяющими количество повторений. Таким образом, данная строка может быть представлена как 4AB5C4D. Длина такой строки 7. Описанный метод мы назовем упаковкой строки.

Напишите программу, которая берет упакованную строчку и восстанавливает по ней исходную строку.

**Входные данные**

Входной файл INPUT.TXT содержит одну упакованную строку. В строке могут встречаться только конструкции вида nA, где n – количество повторений символа (целое число от 2 до 99), а A – заглавная латинская буква, либо конструкции вида A, то есть символ без числа, определяющего количество повторений. Максимальная длина строки не превышает 80.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите восстановленную строку. При этом строка должна быть разбита на строчки длиной ровно по 40 символов (за исключением последней, которая может содержать меньше 40 символов).

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3A4B7D | AAABBBBDDDDDDD |
| 2 | 22D7AC18FGD | DDDDDDDDDDDDDDDDDDDDDDAAAAAAACFFFFFFFFFF<br>FFFFFFFFGD |

| | | |
|---|---|---|
| 3 | 95AB | AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA<br>AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA<br>AAAAAAAAAAAAAAAB |
| 4 | 40AB39A | AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA<br>BAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA |

## 232. Дремучий лес - 2

*(Время: 1 сек. Память: 16 Мб Сложность: 77%)*

Просека – эта такая прямая линия, которая проходит через лес (то есть деревья есть как с одной стороны от этой линии, так и с другой), и при этом она не проходит ни через одно из деревьев леса, а также не касается деревьев. Будем говорить, что лес является дремучим, если в нем нет ни одной просеки.

На плане леса все деревья изображаются кругами. Никакие два круга не пересекаются и не касаются друг друга. Требуется по этому плану определить, является ли лес дремучим.

**Входные данные**

Во входном файле INPUT.TXT содержится сначала целое число N – количество деревьев ($1 \le N \le 200$). Затем идет N троек чисел, задающих деревья. Первые два числа задают координаты центра, а третье – радиус. Все данные задаются точно, и выражаются вещественными числами, не более чем с 2 знаками после десятичной точки, по модулю не превосходящими 1000.

**Выходные данные**

В первой строке выходного файла OUTPUT.TXT должно содержаться сообщение YES, если лес является дремучим, и NO иначе.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3<br>0 10 2<br>5 11 2<br>12.04 7 2 | NO |
| 2 | 3<br>0 0 1<br>2.05 0 1<br>1.02 -1.9 1 | YES |

## 233. Автобусная экскурсия

*(Время: 1 сек. Память: 16 Мб Сложность: 14%)*

Оргкомитет Московской городской олимпиады решил организовать обзорную экскурсию по Москве для участников олимпиады. Для этого был заказан двухэтажный автобус (участников олимпиады достаточно много и в обычный они не умещаются) высотой 437 сантиметров. На экскурсионном маршруте встречаются N мостов. Жюри и оргкомитет олимпиады очень обеспокоены тем, что высокий двухэтажный автобус может не проехать под одним из них. Им удалось выяснить точную высоту каждого из мостов. Автобус может проехать под мостом тогда и только тогда, когда высота моста превосходит высоту автобуса.

Помогите организаторам узнать, закончится ли экскурсия благополучно, а если нет, то установить, где произойдет авария.

**Входные данные**

Во входном файле INPUT.TXT сначала содержится число N ($1 \le N \le 1000$). Далее идут N натуральных чисел, не превосходящих 10000 – высоты мостов в сантиметрах в том порядке, в котором они встречаются на пути автобуса.

**Выходные данные**

В единственную строку выходного файла OUTPUT.TXT нужно вывести фразу «No crash», если экскурсия закончится благополучно. Если же произойдет авария, то нужно вывести сообщение «Crash k», где k – номер моста, где произойдет авария. Фразы выводить без кавычек ровно с одним пробелом внутри.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 1<br>763 | No crash |
| 2 | 3<br>763 245 113 | Crash 2 |
| 3 | 1<br>437 | Crash 1 |

## 234. Сапер

*(Время: 1 сек. Память: 16 Мб Сложность: 28%)*

Мальчику Васе очень нравится известная игра «Сапер» («Minesweeper»).

В «Сапер» играет один человек. Игра идет на клетчатом поле (далее будем называть его картой) NxM (N строк, M столбцов). В K клетках поля стоят мины, в остальных клетках записано либо число от 1 до 8 – количество мин в соседних клетках, либо ничего не написано, если в соседних клетках мин нет. Клетки являются соседними, если они имеют хотя бы одну общую точку, в одной клетке не может стоять более одной мины. Изначально все клетки поля закрыты. Игрок за один ход может открыть какую-нибудь клетку. Если в открытой им клетке оказывается мина – он проигрывает, иначе игроку показывается число, которое стоит в этой клетке, и игра продолжается. Цель игры – открыть все клетки, в которых нет мин.

У Васи на компьютере есть эта игра, но ему кажется, что все карты, которые в ней есть, некрасивые и неинтересные. Поэтому он решил нарисовать свои. Однако фантазия у него богатая, а времени мало, и он хочет успеть нарисовать как можно больше карт. Поэтому он просто выбирает N, M и K и расставляет мины на поле, после чего все остальные клетки могут быть однозначно определены. Однако на определение остальных клеток он не хочет тратить свое драгоценное время. Помогите ему!

По заданным N, M, K и координатам мин восстановите полную карту.

**Входные данные**

В первой строке входного файла INPUT.TXT содержатся числа N, M и K ($1 \le N \le 200$, $1 \le M \le 200$, $0 \le K \le N*M$). Далее идут K строк, в каждой из которых содержится по два числа, задающих координаты мин. Первое число в каждой строке задает номер строки клетки, где находится мина, второе число – номер столбца. Левая верхняя клетка поля имеет координаты (1,1), правая нижняя – координаты (N, M).

**Выходные данные**

Выходной файл OUTPUT.TXT должен содержать N строк по M символов – соответствующие строки карты. j-й символ i-й строки должен содержать символ ‘*‘ (звездочка) если в клетке (i, j) стоит мина, цифру от 1 до 8, если в этой клетке стоит соответствующее число, либо ‘.‘ (точка), если клетка (i, j) пустая.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 10 9 23<br>1 8<br>2 3<br>3 2<br>3 3<br>4 3<br>5 7<br>6 7<br>7 1<br>7 2<br>7 3<br>7 4<br>7 5<br>7 6<br>7 7<br>7 8<br>8 1<br>8 3<br>8 5<br>8 7<br>9 3<br>9 5<br>9 6<br>9 7 | .111..1*1<br>13*2..111<br>1**3.....<br>13*2.111.<br>.111.2*2.<br>233335*41<br>********1<br>*6*7*8*41<br>13*4***2.<br>.1122321. |

## 235. Робот К-79

*(Время: 1 сек. Память: 16 Мб Сложность: 30%)*

Петя написал программу движения робота К-79. Программа состоит из следующих команд:

- S – сделать шаг вперед
- L – повернуться на 90 градусов влево
- R – повернуться на 90 градусов вправо

Напишите программу, которая по заданной программе для робота определит, сколько шагов он сделает прежде, чем впервые вернется на то место, на котором уже побывал до этого, либо установит, что этого не произойдет.

**Входные данные**

Во входном файле INPUT.TXT записана одна строка из заглавных латинских букв S, L, R, описывающая программу для робота. Общее число команд в программе не превышает 200, при этом команд S – не более 50.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите, сколько шагов будет сделано (то есть, выполнено команд S) прежде, чем робот впервые окажется в том месте, через которое он уже проходил. Если такого не произойдет, выведите в выходной файл число –1.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | SSLSLSLSSRSRS | 5 |
| 2 | LSSSS | –1 |

## 236. Многочлен

*(Время: 1 сек. Память: 16 Мб Сложность: 41%)*

Васе задали несколько однотипных задач по математике: «найти значение многочлена». Он хочет написать программу, которая по заданному многочлену и значению x находила бы ответ. Напишите такую программу!

**Входные данные**

В первой строке входного файла INPUT.TXT записан многочлен в виде суммы одночленов. Между одночленами находится знак + или –. Перед первым одночленом может быть знак –. Одночлен записывается как [<Коэффициент>*]x[^<Степень>] или <Коэффициент>, где <Коэффициент> – натуральное число, не превосходящее 100, x – символ переменной (всегда маленькая латинская буква x), <Степень> – натуральное число, не превосходящее 4. Параметры, взятые в квадратные скобки, могут быть опущены. Общее число символов в записи многочлена не превышает 80. Во второй строке записано одно целое число – значение x, не превышающее 100 по абсолютной величине.

**Выходные данные**

В выходной файл OUTPUT.TXT нужно записать одно число – значение данного многочлена при данном значении x.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 8*x+5<br>7 | 61 |
| 2 | -2+x^1-3*x^2+x^2+100*x^3-2*x<br>0 | -2 |

## 237. Головоломка

*(Время: 1 сек. Память: 16 Мб Сложность: 35%)*

| P | O | L | T | E |
|---|---|---|---|---|
| R | W | Y | M | S |
| O | A | I | P | T |
| B | D | A | N | R |
| L | E | M | E | S |

Петя разгадывает головоломку, которая устроена следующим образом. Дана квадратная таблица размера NxN, в каждой клетке которой записана какая-нибудь латинская буква. Кроме того, дан список ключевых слов. Пете нужно, взяв очередное ключевое слово, найти его в таблице. То есть найти в таблице все буквы этого слова, причем они должны быть расположены так, чтобы клетка, в которой расположена каждая последующая буква слова, была соседней с клеткой, в которой записана предыдущая буква (клетки называются соседними, если они имеют общую сторону – то есть соседствуют по вертикали или по горизонтали). Например, на рисунке показано, как может быть расположено в таблице слово olympiad.

Когда Петя находит слово, он вычеркивает его из таблицы. Использовать уже вычеркнутые буквы в других ключевых словах нельзя. После того, как найдены и вычеркнуты все ключевые слова, в таблице остаются еще несколько букв, из которых Петя должен составить слово, зашифрованное в головоломке.

Помогите Пете в решении этой головоломки, написав программу, которая по данной таблице и списку ключевых слов выпишет, из каких букв Петя должен сложить слово, то есть какие буквы останутся в таблице после вычеркивания ключевых слов.

**Входные данные**

Во входном файле INPUT.TXT записаны два числа N ($1 \le N \le 10$) и M ($0 \le M \le 200$). Следующие N строк по N заглавных латинских букв описывают ребус. Следующие M строк содержат слова. Слова состоят только из заглавных латинских букв, каждое слово не длиннее 200 символов. Гарантируется, что в таблице можно найти и вычеркнуть по описанным выше правилам все ключевые слова.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите в алфавитном порядке оставшиеся в таблице буквы.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 5 3<br>POLTE<br>RWYMS<br>OAIPT<br>BDANR<br>LEMES<br>OLYMPIAD<br>PROBLEM<br>TEST | AENRSW |
| 2 | 3 2<br>ISQ<br>ABC<br>IQW<br>I<br>IS | ABCQQW |

## 238. Побег с космической станции

*(Время: 1 сек. Память: 16 Мб Сложность: 52%)*

Представьте, что вы состоите на службе во внешней разведке Межгалактического Альянса Республиканских Сил (МАРС). Одному из агентов разведки крупно не повезло, и он был захвачен на засекреченной космической базе. К счастью, внешней разведке МАРС удалось заполучить план этой базы. И вот теперь вам поручено разработать план побега.

База представляет собой прямоугольник размером NxM, со всех сторон окружённый стенами, и состоящий из квадратных отсеков единичной площади. База снабжена K выходами, до одного из которых агенту необходимо добраться. В некоторых отсеках базы находятся стены. Ваш агент может перемещаться из отсека в любой из четырех соседних с ним, если в том отсеке, куда он хочет переместиться, нет стены.

Кроме того, база снабжена системой гипертуннелей, способных перемещать агента из одного отсека базы (вход в гипертуннель) в другой (выход из гипертуннеля). Когда агент находится в отсеке, где есть вход в гипертуннель, он может (но не обязан) им воспользоваться.

Начальное положение вашего агента известно. Вам необходимо найти кратчайший путь побега (то есть путь, проходящий через минимальное количество отсеков).

**Входные данные**

В первой строке входного файла INPUT.TXT записаны числа N и M ($2 \le N \le 100$, $2 \le M \le 100$), задающие размеры базы: N – количество строк в плане базы, M – количество столбцов. Во второй строке записаны начальные координаты агента $X_A$,$Y_A$ ($1 \le X_A \le N$, $1 \le Y_A \le M$). Первая координата задает номер строки, вторая – номер столбца. Строки нумеруются сверху вниз, столбцы слева направо. Далее следуют N строк по M чисел, задающих описание стен внутри базы: 1 соответствует стенке, 0 – её отсутствию. Далее в отдельной строке записано число H ($0 \le H \le 1000$) – количество гипертуннелей. В последующих H строках идут описания гипертуннелей. Каждый гипертуннель задается 4 числами: $X_1$, $Y_1$, $X_2$, $Y_2$ ($1 \le X_1, X_2 \le N$; $1 \le Y_1, Y_2 \le M$) – координатами входа и выхода гипертуннеля. Никакие два гипертуннеля не имеют общего входа. После этого в отдельной строке следует число K ($1 \le K \le 10$) – количество выходов с базы. В последующих K строках идут описания выходов с базы. Каждый выход задается двумя координатами X и Y ($1 \le X \le N$; $1 \le Y \le M$).

Гарантируется, что начальные координаты агента не совпадают ни с одним из выходов и он не стоит в отсеке, занятом стеной. Никакие входы и выходы гипертуннелей, а также выходы с базы не находятся в отсеках, занятых стенами. Никакой вход в гипертуннель не совпадает с выходом с базы.

**Выходные данные**

Если побег невозможен, выведите в выходной файл OUTPUT.TXT «Impossible». В противном случае следует вывести количество отсеков в кратчайшем пути побега.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 4 5<br>2 1<br>0 0 0 0 0<br>0 1 0 0 0<br>0 0 0 0 0<br>0 0 0 0 0<br>2<br>1 2 1 4<br>3 1 1 4<br>1<br>2 4 | 4 |

## 239. Узор

*(Время: 1 сек. Память: 16 Мб Сложность: 80%)*

В комнате решили сделать паркетный пол. Причем есть задумка выложить на полу некоторый узор. Плитки паркета, которыми выкладывается пол комнаты, состоят из квадратиков 1x1, каждый из которых может быть либо белым, либо черным. В свою очередь, комната имеет размеры NxM. На плане комнаты указано, какой квадрат комнаты какого цвета должен быть. Существует несколько форм паркетных плиток:


Форма 1 Форма 2 Форма 3 Форма 4

Квадратики одной паркетной плитки могут быть окрашены по-разному. Может существовать несколько типов плиток одинаковой формы, но окрашенных по-разному. Плитки разных типов могут иметь разную стоимость. Количество плиток каждого типа не ограничено. Плитки разрешается как угодно поворачивать (на углы, кратные 90 градусам). Не разрешается разламывать плитки, а также класть их лицевой стороной вниз.

Изначально, какая-то часть пола может уже быть выложена плиткой. Требуется определить минимальную стоимость плитки, необходимой для того, чтобы замостить оставшуюся часть комнаты.

**Входные данные**

В первой строке входного файла INPUT.TXT записаны три числа: N, M (размеры комнаты) и K (количество доступных видов плитки). $1 \le N \le 8$, $1 \le M \le 8$, $1 \le K \le 10$. Далее идет описание желаемой раскраски пола. Описание представляет собой N строчек по M чисел в каждой, где 0 обозначает белый цвет, 1 – черный, 2 – то, что квадрат уже выложен плиткой. В последних K строчках находятся описания доступных типов плитки в следующем формате:

<форма> <стоимость> <окраска>

<Форма> – это число от 1 до 4, описывающее форму плитки (см. рисунок выше)

<Стоимость> – это натуральное число, не превосходящее 10000, задающее стоимость одной плитки такого типа

<Окраска> – это от одного до трех чисел 0 или 1. Количество чисел совпадает с количеством квадратиков, из которых состоит плитка. Числа задают цвета квадратиков плитки в том порядке, в каком квадратики пронумерованы на рисунке.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите единственное число – минимальную стоимость укладки или –1, если требуемым образом уложить плитку невозможно.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 4 3 3<br>2 2 2<br>2 0 0<br>2 1 2<br>2 2 2<br>2 10 0 0<br>1 5 1<br>4 6 0 0 1 | 15 |

## 240. Кубическая гостиница

*(Время: 1 сек. Память: 16 Мб Сложность: 60%)*

В связи с проведением межпланетного шашечного турнира было принято решение о строительстве орбитальной гостиницы. Она должна была представлять собой большой куб из N×N×N блоков – маленьких кубиков 1×1×1, и каждый блок должен был быть окрашен снаружи со всех сторон в какой-то один цвет. При этом некоторые блоки могли быть покрашены в один и тот же цвет.

Через год были сделаны фотографии гостиницы с каждой из 6 сторон: спереди, слева, сзади, справа, сверху, снизу. За год эксплуатации могло случиться так, что из-за непрочного крепления некоторые блоки, из которых была построена гостиница, оторвались и улетели в открытый космос. Комиссия по восстановлению гостиницы хочет по сделанным снимкам установить максимальное возможное количество оставшихся блоков.

Итак, вам необходимо по видам гостиницы (куба N×N×N, из которого, возможно, выкинуты некоторые кубики 1×1×1) с 6 сторон определить, из какого максимального количества блоков 1×1×1 она может состоять. Может оказаться так, что гостиница состоит из двух или более не связанных между собой частей.

**Входные данные**

В первой строке входного файла INPUT.TXT находится число N – размер гостиницы ($1 \le N \le 10$). На следующих N строках записаны виды гостиницы с 6 сторон (в следующем порядке: спереди, слева, сзади, справа, сверху, снизу). Каждый такой вид представляет собой таблицу N×N, в которой различными заглавными латинскими буквами обозначены различные цвета, а символом «.» (точка) – то, что в этом месте можно будет смотреть прямо сквозь гостиницу. Два последовательных вида отделяются друг от друга ровно одним пробелом в каждой из N строк.

Нижняя граница вида сверху соответствует верхней границе вида спереди, а верхняя граница вида снизу – нижней границе вида спереди. Для видов спереди, сзади и с боков верх и низ вида соответствуют верху и низу гостиницы.

Входные данные корректны, то есть во входном файле описано состояние, которое может получиться.

**Выходные данные**

Выведите в выходной файл OUTPUT.TXT одно число – искомое максимальное количество оставшихся блоков в гостинице.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3<br>.R. YYR .Y. RYY .Y. .R.<br>GRB YGR BYG RBY GYB GRB<br>.R. YRR .Y. RRY .R. .Y. | 11 |
| 2 | 2<br>ZZ ZZ ZZ ZZ ZZ ZZ<br>ZZ ZZ ZZ ZZ ZZ ZZ | 8 |

## 241. Праздники

*(Время: 1 сек. Память: 16 Мб Сложность: 39%)*

Парламент некоторой страны принял новый закон о праздничных днях. Согласно этому закону первые K дней года, а также 23 февраля и 8 марта объявляются праздничными, а все остальные праздники отменяются. При этом все выходные (суббота и воскресенье), попавшие на праздничные дни, переносятся на следующие за этими праздниками рабочие дни.

В зависимости от того, на какой день недели приходится 1 января, количество нерабочих дней, которые идут подряд, может меняться.

Требуется определить, какое наибольшее количество нерабочих дней может идти подряд.

**Входные данные**

Во входном файле INPUT.TXT записано единственное число K ($1 \le K \le 50$).

**Выходные данные**

В выходной файл OUTPUT.TXT требуется записать единственное число – наибольшее количество нерабочих дней, идущих подряд.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 2 | 4 |
| 2 | 10 | 16 |

## 242. Раскраска плиток

*(Время: 1 сек. Память: 16 Мб Сложность: 66%)*

После того, как к удивлению тётушки Полли, её забор был покрашен, она поручила Тому Сойеру обновить краску на плитках, которыми был вымощен их квадратный двор. Двор был покрыт NxN одинаковыми квадратными плитками, каждая из которых когда-то давно была покрашена в один из K цветов ($K < N$). Краска на плитках потускнела и Тому Сойеру поручили их покрасить, на этот раз в один любой цвет (из тех же K цветов). Покрасить нужно все плитки, в том числе и те, которые уже были покрашены в этот цвет раньше.

Окунув кисть в ведро с краской один раз, можно перекрасить один горизонтальный или вертикальный ряд плиток. Чтобы разнообразить свою работу, Том придумал, что ряд плиток можно красить только цветом, которым на данный момент уже покрашены (старой или новой краской) по крайней мере две плитки выбранного ряда (вертикального или горизонтального). За один раз Том собирается красить допустимым цветом весь ряд целиком, независимо от того, были ли уже перекрашены какие-либо его плитки ранее. Помогите Тому определить, какое минимальное число раз ему придется обмакнуть кисть, чтобы перекрасить все плитки, следуя придуманным правилам, и в какой цвет окажутся окрашены все плитки.

**Входные данные**

В первой строке входного файла INPUT.TXT записаны через пробел два числа: N – количество плиток в одном ряду ($1 < N \le 200$) и K ($1 \le K < N$). В каждой из следующих N строк записаны N натуральных чисел, обозначающих номера цветов красок, в которые когда-то были выкрашены соответствующие плитки данного горизонтального ряда. Номера цветов – натуральные числа в диапазоне от 1 до K.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите два числа: L – какое минимальное число раз придется окунать кисть в ведро с краской, и номер краски C, в которую в результате окажутся перекрашены все плитки двора. Если таких красок может быть несколько, то выведите наименьший номер. Если перекрасить все плитки, следуя придуманным Томом правилам, нельзя, выведите два раза число 0.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3 2<br>1 2 1<br>2 1 1<br>1 2 2 | 4 1 |
| 2 | 2 1<br>1 1<br>1 1 | 2 1 |

## 243. Маскарад

*(Время: 1 сек. Память: 16 Мб Сложность: 63%)*

По случаю введения больших новогодних каникул устраивается великий праздничный бал-маскарад. До праздника остались считанные дни, поэтому срочно нужны костюмы для участников. Для пошивки костюмов требуется L метров ткани. Ткань продается в N магазинах, в которых предоставляются скидки оптовым покупателям. В магазинах можно купить только целое число метров ткани. Реклама магазина номер i гласит «Мы с радостью продадим Вам метр ткани за Pi рублей, однако если Вы купите не менее Ri метров, то получите прекрасную скидку – каждый купленный метр обойдется Вам всего в Qi рублей». Чтобы воплотить в жизнь лозунг «экономика страны должна быть экономной», правительство решило потратить на закупку ткани для костюмов минимальное количество рублей из государственной казны. При этом ткани можно купить больше, чем нужно, если так окажется дешевле. Ответственный за покупку ткани позвонил в каждый магазин и узнал, что:

1. реклама каждого магазина содержит правдивую информацию о ценах и скидках;
2. магазин номер i готов продать ему не более Fi метров ткани.

Ответственный за покупку очень устал от проделанной работы и поэтому поставленную перед ним задачу «закупить ткань за минимальные деньги» переложил на своих помощников. Напишите программу, которая определит, сколько ткани нужно купить в каждом из магазинов так, чтобы суммарные затраты были минимальны.

**Входные данные**

В первой строке входного файла записаны два целых числа N и L ($1 \le N \le 100$, $0 \le L \le 100$). В каждой из последующих N строк находится описание магазина номер i – 4 целых числа Pi, Ri, Qi, Fi ($1 \le Qi \le Pi \le 1000$, $1 \le Ri \le 100$, $0 \le Fi \le 100$).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите минимально возможную стоимость материи. Если покупка невозможна, выведите -1.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 2 14<br>7 9 6 10<br>7 8 6 10 | 88 |
| 2 | 1 20<br>1 1 1 1 | -1 |

## 244. Билетики

*(Время: 1 сек. Память: 16 Мб Сложность: 37%)*

В процессе установки турникетов в автобусах, разработчики столкнулись с проблемой проверки подлинности билета. Для ее решения был придуман следующий способ защиты от подделок.

Информация, записанная на билете, кодируется K числами (0 или 1). При этом непосредственно на билете записывается последовательность из N чисел ($N \ge K$) так, что числа, записанные на расстоянии K, совпадают. Таким образом, для проверки подлинности билета достаточно проверить, что все числа на расстоянии K совпадают. К сожалению, при считы-

вании информации с билета иногда могут происходить ошибки – считается, что одно из чисел может исказиться (то есть 0 замениться на 1, или 1 – на 0). Такой билет все равно нужно считать подлинным. Во всех остальных случаях билет считается поддельным.

Напишите программу, которая по информации, считанной с билета, устанавливает его подлинность, и указывает на то, при считывании какого из чисел могла произойти ошибка.

**Входные данные**

В первой строке входного файла INPUT.TXT записаны числа N и K ($1 \le N \le 50000$, $1 \le K \le 1000$, $K \le N$). Во второй строке записано N чисел, каждое из которых является 0 или 1 – информация, считанная с билета.

**Выходные данные**

В первой строке выходного файла OUTPUT.TXT должно быть записано OK, если билет подлинный и FAIL, если поддельный. В случае, если билет подлинный, во второй строке выведите 0, если все числа были считаны правильно, или номер числа, в котором при считывании произошла ошибка. Если возможных искаженных номеров несколько, выведите наименьший из них.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 6 2<br>1 0 1 0 1 0 | OK<br>0 |
| 2 | 8 5<br>0 1 1 0 1 0 0 1 | OK<br>2 |
| 3 | 6 2<br>1 1 1 0 0 0 | FAIL |

## 245. Сплоченная команда

*(Время: 1 сек. Память: 16 Мб Сложность: 42%)*

Как показывает опыт, для создания успешной футбольной команды важны не только умения отдельных её участников, но и сплочённость команды в целом. Характеристикой умения игрока является показатель его профессионализма (ПП). Команда является сплочённой, если ПП каждого из игроков не превосходит суммы ПП любых двух других (в частности, любая команда из одного или двух игроков является сплоченной). Перед тренерским составом молодёжной сборной была поставлена задача сформировать сплоченную сборную с максимальной суммой ПП игроков (ограничений на количество игроков в команде нет).

Ваша задача состоит в том, чтобы помочь сделать правильный выбор из N человек, для каждого из которых известен его ПП.

**Входные данные**

В первой строке входного файла INPUT.TXT записано целое число N ($0 \le N \le 30000$). В последующих N строках записано по одному целому числу Pi ($0 \le Pi \le 60000$), представляющему собой ПП соответствующего игрока.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите максимально возможную сумму ПП игроков сплоченной команды.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 4<br>1<br>5<br>3<br>3 | 11 |

| | | |
|---|---|---|
| 2 | 5<br>100<br>20<br>20<br>20<br>20 | 120 |

## 246. Вагоны

*(Время: 1 сек. Память: 16 Мб Сложность: 28%)*

Ежедневно диспетчеру железнодорожной станции приходится переставлять вагоны во многих поездах, чтобы они шли в заданном порядке. Для этого диспетчер может расцепить пришедший на станцию состав в произвольных местах и переставить образовавшиеся сцепки из одного или нескольких вагонов в произвольном порядке. Порядок вагонов в одной сцепке менять нельзя, также нельзя развернуть всю сцепку так, чтобы последний вагон в сцепке оказался первым в ней.

Диспетчер просит вашей помощи в определении того, какое минимальное число соединений между вагонами необходимо расцепить, чтобы переставить вагоны в составе в требуемом порядке.

**Входные данные**

В первой строке входного файла INPUT.TXT содержится целое число N ($1 \le N \le 100$). Во второй строке содержится перестановка натуральных чисел от 1 до N (то есть все натуральные числа от 1 до N в некотором порядке). Числа разделяются одним пробелом. Эта перестановка задает номера вагонов в приходящем на станцию составе. Требуется, чтобы в уходящем со станции составе вагоны шли в порядке их номеров.

**Выходные данные**

Программа должна записать в выходной файл OUTPUT.TXT единственное целое число, равное минимальному количеству соединений между вагонами, которое нужно расцепить в данном составе, чтобы их можно было переставить по порядку.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 4<br>3 1 2 4 | 2 |
| 2 | 5<br>5 4 3 2 1 | 4 |
| 3 | 2<br>1 2 | 0 |

## 247. Кафе

*(Время: 1 сек. Память: 16 Мб Сложность: 46%)*

Около Петиного университета недавно открылось новое кафе, в котором действует следующая система скидок: при каждой покупке более чем на 100 рублей покупатель получает купон, дающий право на один бесплатный обед (при покупке на сумму 100 рублей и меньше такой купон покупатель не получает).

Однажды Пете на глаза попался прейскурант на ближайшие N дней. Внимательно его изучив, он решил, что будет обедать в этом кафе все N дней, причем каждый день он будет покупать в кафе ровно один обед. Однако стипендия у Пети небольшая, и поэтому он хочет по максимуму использовать предоставляемую систему скидок так, чтобы его суммарные затраты были минимальны. Требуется найти минимально возможную суммарную стоимость обедов и номера дней, в которые Пете следует воспользоваться купонами.

**Входные данные**

В первой строке входного файла INPUT.TXT записано целое число N ($0 \le N \le 100$). В каждой из последующих N строк записано одно целое число, обозначающее стоимость обеда в рублях на соответствующий день. Стоимость – неотрицательное целое число, не превосходящее 300.

**Выходные данные**

В первой строке выходного файла OUTPUT.TXT выдайте минимальную возможную суммарную стоимость обедов. Во второй строке выдайте два числа K1 и K2 – количество купонов, которые останутся неиспользованными у Пети после этих N дней и количество использованных им купонов соответственно. Если существует несколько решений с минимальной суммарной стоимостью, то выдайте то из них, в котором значение K1 максимально (на случай, если Петя когда-нибудь ещё решит заглянуть в это кафе).

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 5<br>35<br>40<br>101<br>59<br>63 | 235<br>0 1 |

## 248. EuroEnglish

*(Время: 1 сек. Память: 16 Мб Сложность: 55%)*

Европейская комиссия планирует принять решение о том, что официальным языком Евросоюза станет английский. Был также разработан план упрощения английской письменности, который планируется реализовать за четыре года.

Первоочередной задачей будет избавление от буквы c, которая в сочетаниях ci и ce будет изменяться на s, в сочетании ck – опускаться, а в остальных случаях заменяться на k. При этом все замены будут производиться в строгом порядке слева направо. То есть, например, в слове success сначала первая из двух букв c заменится на k, а затем вторая – на s, то есть получится suksess. А слово cck превратится в kk.

На второй год из английских слов изымут все удвоенные буквы: ee изменят на i, oo – на u, а в остальных комбинациях будут просто писать одну букву вместо двух одинаковых. Такие замены также будут делать строго в порядке слева направо. Так, слово ooo превратится в uo, а oou – просто в u (в нем сначала oo заменится на u, а затем uu – на u), слово iee превратится в i (в нем сначала ee заменится на i, а затем ii – на i).

На третий год на конце слова станут опускать букву e, если она не единственная буква в слове.

Наконец, завершением реформы станет отмена артиклей (в английском языке три артикля: a, an и the). При этом удаляться эти артикли будут только тогда, когда они в исходном тексте были словами a, an, the. То есть, например, текст the table после реформ первых трех лет превратиться в th tabl, а после реформы четвертого года – просто в tabl. А слово aaaaa после реформы первых лет станет словом a, но поскольку изначально оно не было словом a (артиклем), то оно в итоге так и останется словом a.

Напишите программу, которая будет переводить классический английский текст на Евроинглиш.

**Входные данные**

Во входном файле INPUT.TXT записана одна строка текста, состоящая не более чем из 200 символов: латинских строчных и заглавных букв, пробелов и знаков препинания (в тексте могут встречаться: точка, запятая, вопросительный и восклицательный знаки, двоеточие, тире, точка с запятой, открывающаяся и закрывающаяся скобки, апострофы, кавычки). Заглавные буквы могут встречаться только в начале слова. Нигде подряд не могут стоять два пробела. В начале и в конце строки не может стоять пробел. Слова отделяются друг от друга пробелами и/или знаками препинания.

**Выходные данные**

В выходной файл OUTPUT.TXT нужно выдать преобразованную строку при ограничениях:

- начинаться с заглавной буквы должны те и только те слова, которые начинались с заглавной буквы в исходном тексте;

- не должно встречаться двух пробелов подряд;
- пробелы между словами и знаками препинания должны остаться там и только там, где они были в исходной строке, в начале и в конце строки пробелов быть не должно.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | `cacao and coffee` | `kakao and kofi` |
| 2 | `Cinderella! Where Is The Dress???` | `Sinderela! Wher Is Dres???` |
| 3 | `'A' is a letter` | `'' is leter` |
| 4 | `!!!Hello!!!A-the-"word"` | `!!!Helo!!!--"word"` |
| 5 | `Aaaa then the ckckck` | `A then k` |
| 6 | `"A"-the an` | `""-` |
| 7 | `A the an` | |
| 8 | `success` | `sukses` |

## 249. Скобки

*(Время: 1 сек. Память: 16 Мб Сложность: 61%)*

Назовем строку S правильной скобочной последовательностью, если она состоит только из символов '{', '}', '[', ']', '(', ')' и выполнено хотя бы одно из следующих трех условий:

1. S – пустая строка;
2. S можно представить в виде $S=S_1+S_2+S_3+...+S_N$ (N>1), где $S_i$ – непустые правильные скобочные последовательности, а знак "+" обозначает конкатенацию (приписывание) строк;
3. S можно представить в виде S='{'+C+'}' или S='['+C+']' или S='('+C+')', где C является правильной скобочной последовательностью.

Дана строка, состоящая только из символов '{', '}', '[', ']', '(', ')'. Требуется определить, какое минимальное количество символов надо вставить в эту строку для того, чтобы она стала правильной скобочной последовательностью.

**Входные данные**

В первой строке входного файла INPUT.TXT записана строка, состоящая только из символов '{', '}', '[',']', '(', ')'. Длина строки не превосходит 100 символов.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите ответ на поставленную задачу.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | `{()}` | `2` |
| 2 | `([{}])` | `0` |

## 250. Двоякие числа

*(Время: 1 сек. Память: 16 Мб Сложность: 27%)*

Натуральное число называется двояким, если в его десятичной записи встречается не более двух различных цифр. Например, числа 3, 23, 33, 100, 12121 – двоякие, а числа 123 и 9980 – нет.

Для заданного натурального числа N требуется найти ближайшее к нему двоякое число (если таких чисел два – любое из них).

**Входные данные**

Во входном файле INPUT.TXT записано одно натуральное число N, не превосходящее 30000.

**Выходные данные**

В выходной файл OUTPUT.TXT требуется выдать единственное число – ближайшее двоякое к числу N. Если таких чисел несколько, то следует вывести наименьшее.

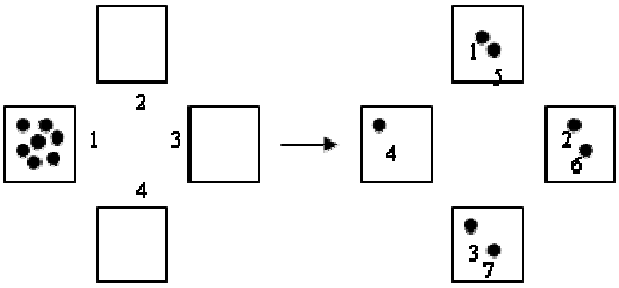
**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 123 | 122 |
| 2 | 2012 | 2020 |
| 3 | 11111 | 11111 |

## 251. Калах

*(Время: 1 сек. Память: 16 Мб Сложность: 57%)*

Для игры в калах используют несколько коробочек, расставленных по кругу, в которых лежат шарики. Ход осуществляется следующим образом. Берутся все шарики из одной коробочки, и начинают раскладываться по одному в коробочки подряд начиная со следующей по часовой стрелке. Если шариков больше, чем коробочек, то процесс продолжается (шарики раскладываются по второму кругу, по третьему и т.д.), пока не будут разложены все шарики. В коробочку, из которой взяли шарики, их тоже кладут. Пример одного хода приведен на рисунке. Справа шарики пронумерованы в том порядке, в котором они раскладывались по коробочкам.




Петя, тренируясь перед соревнованиями, разложил шарики по коробочкам произвольным образом, и стал делать произвольные ходы. После каждого хода он записывал номер коробочки, в которую попадал последний шарик. В некоторый момент он решил восстановить начальную конфигурацию по конечной и по тем записям, которые он делал. Напишите программу, которая поможет ему в этом.

**Входные данные**

В первой строке входного файла INPUT.TXT записано два натуральных числа: $N \le 100$ – количество коробочек и $M \le 100$ – количество сделанных Петей ходов. Коробочки пронумерованы последовательно по часовой стрелке числами от 1 до N. В следующих N строках (либо в одной строке через пробел) записано количество шариков в первой, второй, …, N-ой коробочках в конечной конфигурации. В следующих M строках (либо в одной строке через пробел) записаны номера коробочек, в которые был положен последний шарик на первом, втором, ..., M-ом ходу соответственно. Общее количество шариков не превосходит $10^9$.

**Выходные данные**

В выходной файл OUTPUT.TXT требуется вывести N чисел: первоначальное количество шариков в первой, второй, ..., N-ой коробочках.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 4 1<br>1 2 2 2<br>4 | 7 0 0 0 |
| 2 | 2 2<br>1<br>1<br>2<br>2 | 1 1 |

## 252. Сортировка масс

*(Время: 1 сек. Память: 16 Мб Сложность: 36%)*

Как известно, Россия является одним из ведущих экспортеров нефти. Разные страны мира, от достаточно больших до сравнительно маленьких, нуждаются в этой нефти как в воздухе. В ее состав в больших количествах входят ароматические углеводороды, которые

обуславливают ее высокое качество. Доставка нефти в пункт назначения осуществляется с помощью нефтепровода. Считается, что количество нефти, отправленное в страну назначения, равно количеству полученной нефти. На самом деле это, конечно, не так. Как и многое другое, нефть воруют некоторые несознательные личности. Причем неофициально считается, что больше нефти воруют в нефтепроводах тех стран, куда нефти посылается больше (может быть, несознательные личности считают, что приносят, таким образом, меньше ущерба, кто знает...). Официальное руководство компании «Русская Нефть» решило узнать, правдивый это слух или нет, чтобы усилить (а может просто установить) охрану на тех нефтепроводах, где больше всего воруют нефть.

Для этого им нужно отсортировать нефтепроводы по количеству нефти, которая протекает в направлении какой-то страны за сутки. У компании «Русская Нефть», как и у любой уважающей себя компании, есть несколько штатных программистов, и руководство предложило им решить эту, в сущности, нетрудную задачу. Но программистов поставило в тупик то, что данные о количестве нефти представлены в разных единицах измерения (начиная от граммов и заканчивая тоннами).

Поэтому они решили найти человека, который был бы в силах решить эту задачу за них, и обещают взять его на работу в эту перспективную и процветающую компанию. Решите задачу, и, кто знает, может, повезет именно Вам?

**Входные данные**

В первой строке входного файла INPUT.TXT находится целое число N ($1 \le N \le 1000$) – количество нефтепроводов. В каждой из следующих N строк находится количество (точнее – масса) нефти, транспортированной по соответствующему нефтепроводу за сутки, по одному в строке. Масса нефти задана целым числом от 1 до 10000 с указанием соответствующей единицы измерения. Число и единица измерения разделены ровно одним пробелом. Единица измерения задается одной из трех букв: g (граммы), p (пуды), t (тонны), причем перед этой буквой может стоять одна из приставок: m (милли-), k (кило-), M (мега-), G (гига-). Напомним, что эти приставки обозначают умножение единицы измерения на $10^{-3}$, $10^3$, $10^6$ и $10^9$ соответственно. 1 пуд = 16380 граммов, 1 тонна = $10^6$ граммов.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите N строк, в которых должны быть записаны массы нефти в порядке неубывания. Каждая строка должна описывать массу нефти в одном из нефтепроводов. Массы должны быть описаны в том же формате, в котором записаны во входном файле. Приоритет равных масс, записанных в разных форматах должен соответствовать порядку, в котором они следуют во входном файле.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 5<br>234 g<br>4576 mp<br>2 t<br>32 mg<br>2 Mg | 32 mg<br>234 g<br>4576 mp<br>2 t<br>2 Mg |

## 253. Часы с боем

*(Время: 1 сек. Память: 16 Мб Сложность: 25%)*

Старинные часы бьют каждые полчаса. Причем в начале каждого часа они бьют столько раз, сколько сейчас часов (по 1 разу – в час ночи и в час дня, по 2 раза – в два часа ночи и в два часа дня и т.д., в полночь и в полдень они бьют, соответственно, по 12 раз). И еще 1 раз они бьют в середине каждого часа.

Дан промежуток времени (известно, что прошло строго меньше 24 часов). Напишите программу, определяющую, сколько ударов сделали часы за это время.

**Входные данные**

В первой строке входного файла INPUT.TXT записан начальный момент времени, во второй строке – конечный. Моменты времени задаются двумя целыми числами, разделяющимися пробелом. Первое число задает часы (от 0 до 23), второе – минуты (от 1 до 59, при этом оно не равно 30).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите одно число – сколько ударов сделали часы за этот отрезок времени.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 5 20<br>10 25 | 45 |
| 2 | 10 25<br>5 20 | 135 |
| 3 | 5 2<br>5 21 | 0 |

## 254. Выборы жрецов

*(Время: 1 сек. Память: 16 Мб Сложность: 28%)*

В стране Олимпиадии снова выборы.

Страна состоит из маленьких графств. Графства объединяются в конфедерации. Каждая конфедерация раз в год выбирает себе покровителя – одного из 200 жрецов. Этот ритуал называется Великими Перевыборами Жрецов и выглядит так: конфедерации одновременно подают заявления (одно от конфедерации) в Совет Жрецов о том, кого они хотели бы видеть своим покровителем (если заявление не подано, то считают, что конфедерация хочет оставить себе того же покровителя). После этого все заявки удовлетворяются. Если несколько конфедераций выбирают одного и того же Жреца, то они навсегда объединяются в одну. Таким образом, каждый Жрец всегда является покровителем не более чем одной конфедерации. Требуется написать программу, позволяющую Совету Жрецов выяснить номер Жреца-покровителя каждого графства после Великих Перевыборов. В Совете все графства занумерованы (начиная с 1). Все Жрецы занумерованы числами от 1 до 200 (некоторые из них сейчас могут не быть ничьими покровителями).

**Входные данные**

Во входном файле INPUT.TXT записано число N – количество графств в стране ($1 \le N \le 5000$) – и далее для каждого графства записан номер Жреца-покровителя конфедерации, в которую оно входит (графства считаются по порядку их номеров). Затем указаны заявления от конфедераций. Сначала записано число M – количество поданных заявлений, а затем M пар чисел: первое число – номер текущего Жреца-покровителя, второе – номер желаемого Жреца-покровителя.

Все числа во входном файле разделяются пробелами и (или) символами перевода строки.

**Выходные данные**

В выходной файл OUTPUT.TXT вывести для каждого графства одно число – номер его Жреца-покровителя после Великих Перевыборов. Сначала – для первого графства, затем – для второго и т.д.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 7<br>1 1 5 3 1 5 1<br>2<br>5 1<br>1 3 | 3 3 1 3 3 1 3 |

## 255. Представление чисел

*(Время: 1 сек. Память: 16 Мб Сложность: 40%)*

Дано натуральное число N. Требуется представить его в виде суммы двух натуральных чисел A и B таких, что НОД (наибольший общий делитель) чисел A и B – максимален.

**Входные данные**

Во входном файле INPUT.TXT записано натуральное число N ($2 \le N \le 10^9$).

**Выходные данные**

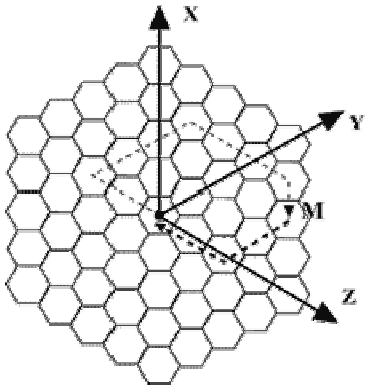
В выходной файл OUTPUT.TXT выведите два искомых числа A и B. Если решений несколько, выведите любое из них.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 15 | 5 10 |
| 2 | 16 | 8 8 |

## 256. Гексагон

*(Время: 1 сек. Память: 16 Мб Сложность: 38%)*




Поле для игры в новую игру «Гексагон» разбито на шестиугольники (см. рисунок). Игрок, стартуя из некоторого начального шестиугольника, сделал несколько ходов. Каждый ход заключается в перемещении фишки в соседний шестиугольник (имеющий с тем, где находилась фишка до начала хода, общую сторону) – тем самым, ход делается вдоль одного из направлений X, Y или Z (см. рисунок). Игрок записал все свои ходы, причем если фишка двигалась вдоль какого-либо направления несколько раз подряд, то в записи это обозначается указанием направления и количества ходов, которые были сделаны.

Напишите программу, которая найдет кратчайший (по количеству совершаемых ходов) путь в начальную клетку из той, где фишка оказалась после ходов игрока.

**Входные данные**

В первой строке входного файла INPUT.TXT записано число N – количество строк в записи перемещений фишки ($1 \le N \le 100$). Далее идет N строк с записью ходов: в каждой строке записана сначала большая буква X, Y или Z, задающая направление, затем пробел, и число, задающее количество ходов в данном направлении (число может быть и отрицательным, если игрок перемещал фишку параллельно оси, но в направлении, противоположном направлению оси). Все числа по модулю не превышают 200.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите длину кратчайшего пути обратно в начальную клетку.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 4<br>Z -2<br>Y 3<br>Z 3<br>X -1 | 4 |

## 257. Кубическое уравнение

*(Время: 1 сек. Память: 16 Мб Сложность: 56%)*

Напишите программу, которая будет искать все целые X, удовлетворяющие уравнению $A*X^3 + B*X^2 + C*X + D = 0$, где A, B, C, D – заданные целые коэффициенты.

**Входные данные**

Во входном файле INPUT.TXT записаны четыре целых числа: A, B, C, D. Все числа по модулю не превышают $2*10^9$.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите сначала количество решений этого уравнения в целых числах, а затем сами корни в возрастающем порядке. Если уравнение имеет бесконечно много корней, то следует вывести в выходной файл одно число -1 (минус один).

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 1 0 0 -27 | 1 3 |
| 2 | 0 1 2 3 | 0 |

## 258. Скорая помощь

*(Время: 1 сек. Память: 16 Мб Сложность: 42%)*

Бригада скорой помощи выехала по вызову в один из отдаленных районов. К сожалению, когда диспетчер получил вызов, он успел записать только адрес дома и номер квартиры K1, а затем связь прервалась. Однако он вспомнил, что по этому же адресу дома некоторое время назад скорая помощь выезжала в квартиру K2, которая расположена в подъезда P2 на этаже N2. Известно, что в доме M этажей и количество квартир на каждой лестничной площадке одинаково. Напишите программу, которая вычисляет номер подъезда P1 и номер этажа N1 квартиры K1.

**Входные данные**

Во входном файле INPUT.TXT записаны пять положительных целых чисел K1, M, K2, P2, N2. Все числа не превосходят 1000.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите два числа P1 и N1. Если входные данные не позволяют однозначно определить P1 или N1, вместо соответствующего числа напечатайте 0. Если входные данные противоречивы, то следует вывести два числа –1 (минус один).

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 89 20 41 1 11 | 2 3 |
| 2 | 11 1 1 1 1 | 0 1 |
| 3 | 3 2 2 2 1 | -1 -1 |

## 259. A-функция от строчки

*(Время: 1 сек. Память: 16 Мб Сложность: 59%)*

Дана строка S, состоящая из N символов. Определим функцию A(i) от первых i символов этой сроки следующим образом:

A(i) = максимально возможному k, что равны следующие строки:

$$S[1]+S[2]+S[3]+\ldots+S[k],$$
$$S[i]+S[i-1]+S[i-2]+\ldots+S[i-k+1],$$

где S[i] – i-ый символ строки S, а знак + означает, что символы записываются в строчку непосредственно друг за другом.

Напишите программу, которая вычислит значения функции A для заданной строчки для всех возможных значений i от 1 до N.

**Входные данные**

В первой строке входного файла INPUT.TXT записано одно число N ($1 \le N \le 200000$). Во второй строке записана строка длиной N символов, состоящая только из больших и/или маленьких латинских букв.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите N чисел – значения функции A(1), A(2), … A(N).

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 5<br>aabaa | 1 2 0 1 5 |

## 260. Олимпиада по алхимии

*(Время: 1 сек. Память: 16 Мб Сложность: 45%)*

В государстве алхимиков есть N населённых пунктов, пронумерованных числами от 1 до N, и M дорог. Населённые пункты бывают двух типов: деревни и города. Кроме того, в государстве есть одна столица (она может располагаться как в городе, так и в деревне). Каждая дорога соединяет два населённых пункта, и для проезда по ней требуется Ti минут. В столице было решено провести 1-ю государственную командную олимпиаду по алхимии. Для этого во все города из столицы были отправлены гонцы (по одному на город) с информацией про олимпиаду.

Напишите программу, которая посчитает, в каком порядке и через какое время каждый из гонцов доберётся до своего города. Считается, что гонец во время пути нигде не задерживается.

**Входные данные**

Во входном файле INPUT.TXT сначала записаны 3 числа N, M, K – количество населенных пунктов, количество дорог и количество городов ($2 \le N \le 1000$, $1 \le M \le 10000$, $1 \le K \le N$). Далее записан номер столицы C ($1 \le C \le N$). Следующие K чисел задают номера городов. Далее следуют M троек чисел Si, Ei, Ti, описывающих дороги: Si и Ei – номера населенных пунктов, которые соединяет данная дорога, а Ti – время для проезда по ней ($1 \le Ti \le 100$).

Гарантируется, что до каждого города из столицы можно добраться по дорогам (возможно, через другие населенные пункты).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите K пар чисел: для каждого города должен быть выведен его номер и минимальное время, когда гонец может в нем оказаться (время измеряется в минутах с того момента, как гонцы выехали из столицы). Пары в выходном файле должны быть упорядочены по времени прибытия гонца, а в случае совпадения времени следует сортировать эти пары по номерам городов.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 5 4 5 1<br>1 2 3 4 5<br>1 2 1<br>2 3 10<br>3 4 100<br>4 5 100 | 1 0<br>2 1<br>3 11<br>4 111<br>5 211 |
| 2 | 5 5 3 1<br>2 4 5<br>2 1 1<br>2 3 10<br>3 4 100<br>4 5 100<br>1 5 1 | 2 1<br>5 1<br>4 101 |

## 261. Лотерея

*(Время: 5 сек. Память: 16 Мб Сложность: 71%)*

На одном из телеканалов каждую неделю проводится следующая лотерея. В течение недели участники делают свои ставки. Каждая ставка заключается в назывании какого-либо M-значного числа в системе счисления с основанием K (то есть, по сути, каждый участник называет M цифр, каждая из которых лежит в диапазоне от 0 до K–1). Ведущие нули в числах допускаются.

В некоторый момент прием ставок на текущий розыгрыш завершается, и после этого ведущий в телеэфире называет выигравшее число (это также M-значное число в K-ичной системе счисления). После этого те телезрители, у кого первая цифра их числа совпала с первой цифрой числа, названного ведущим, получают выигрыш в размере $A_1$ рублей. Те, у кого совпали первые две цифры числа – получают $A_2$ рублей (при этом если у игрока совпала вторая цифра, но не совпала первая, он не получает ничего). Аналогично угадавшие первые три цифры получают $A_3$ рублей. И так далее. Угадавшие все число полностью получают $A_M$ рублей. При этом если игрок угадал t первых цифр, то он получает $A_t$ рублей, но не получает призы за угадывание t–1, t–2 и т.д. цифр. Если игрок не угадал первую цифру, он не получает ничего.

Напишите программу, которая по известным ставкам, сделанным телезрителями, находит число, которое должна назвать телеведущая, чтобы фирма-организатор розыгрыша выплатила в качестве выигрышей минимальную сумму. Для вашего удобства ставки, сделанные игроками, уже упорядочены по неубыванию.

**Входные данные**

В первой строке входного файла INPUT.TXT задаются числа N (количество телезрителей, сделавших свои ставки, $1 \le N \le 100000$), M (длина чисел $1 \le M \le 10$) и K (основание системы счисления $2 \le K \le 10$). В следующей строке записаны M чисел $A_1$, $A_2$, …, $A_M$, задающих выигрыши в случае совпадения только первой, первых двух,... , всех цифр ($1 \le A_1 \le A_2 \le \ldots \le A_M \le 100000$). В каждой из следующих N строк либо в одной строке через пробел записано по одному M-значному K-ичному числу. Числа идут в порядке неубывания.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите наименьшую сумму, которую придется выплатить в качестве выигрыша.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 10 3 2<br>1 3 100<br>000 000 001 010 100 100 100 100 110 111 | 6 |
| 2 | 1 1 10<br>100<br>0 | 0 |

## 262. Коммерческий калькулятор

*(Время: 1 сек. Память: 16 Мб Сложность: 31%)*

Фирма OISAC выпустила новую версию калькулятора. Этот калькулятор берет с пользователя деньги за совершаемые арифметические операции. Стоимость каждой операции в долларах равна 5% от числа, которое является результатом операции.

На этом калькуляторе требуется вычислить сумму N натуральных чисел (числа известны). Нетрудно заметить, что от того, в каком порядке мы будем складывать эти числа, иногда зависит, в какую сумму денег нам обойдется вычисление суммы чисел (тем самым, оказывается нарушен классический принцип «от перестановки мест слагаемых сумма не меняется» ☺).

Например, пусть нам нужно сложить числа 10, 11, 12 и 13. Тогда если мы сначала сложим 10 и 11 (это обойдется нам в \$1.05), потом результат – с 12 (\$1.65), и затем – с 13 (\$2.3), то всего мы заплатим \$5, если же сначала отдельно сложить 10 и 11 (\$1.05), потом – 12 и 13 (\$1.25) и, наконец, сложить между собой два полученных числа (\$2.3), то в итоге мы заплатим лишь \$4.6.

Напишите программу, которая будет определять, за какую минимальную сумму денег можно найти сумму данных N чисел.

**Входные данные**

Во входном файле INPUT.TXT записано число N ($2 \le N \le 100000$). Далее идет N натуральных чисел, которые нужно сложить, каждое из них не превышает 10000.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите, сколько денег нам потребуется на нахождение суммы этих N чисел. Результат должен быть выведен с двумя знаками после десятичной точки без лидирующих нулей.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 4<br>10 11 12 13 | 4.60 |
| 2 | 2<br>1 1 | 0.10 |

## 263. Метро

*(Время: 1 сек. Память: 16 Мб Сложность: 16%)*

Витя работает недалеко от одной из станций кольцевой линии метро, а живет рядом с другой станцией той же линии. Требуется выяснить, мимо какого наименьшего количества промежуточных станций необходимо проехать Вите по кольцу, чтобы добраться с работы домой.

**Входные данные**

Во входном файле INPUT.TXT заданы три числа: сначала N – общее количество станций кольцевой линии, а затем i и j – номера станции, на которой Витя садится, и станции, на которой он должен выйти. Станции пронумерованы подряд натуральными числами 1, 2, 3, …, N (1-я станция – соседняя с N-й), N не превосходит 100. Числа i и j не совпадают. Все числа разделены пробелом.

**Выходные данные**

В выходной файл OUTPUT.TXT требуется вывести минимальное количество промежуточных станций (не считая станции посадки и высадки), которые необходимо проехать Вите.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 100 5 6 | 0 |
| 2 | 10 1 9 | 1 |

## 264. Оттепель

*(Время: 1 сек. Память: 16 Мб Сложность: 17%)*

Уставшие от необычно теплой зимы, жители решили узнать, действительно ли это самая длинная оттепель за всю историю наблюдений за погодой. Они обратились к синоптикам, а те, в свою очередь, занялись исследованиями статистики за прошлые годы. Их интересует, сколько дней длилась самая длинная оттепель.

Оттепелью они называют период, в который среднесуточная температура ежедневно превышала 0 градусов Цельсия. Напишите программу, помогающую синоптикам в работе.

**Входные данные**

Во входном файле INPUT.TXT сначала записано число N – общее количество рассматриваемых дней ($1 \le N \le 100$). В следующей строке через пробел располагается N целых чисел, разделенных пробелами. Каждое число – среднесуточная температура в соответствующий день. Температуры – целые числа и лежат в диапазоне от –50 до 50.

**Выходные данные**

В выходной файл OUTPUT.TXT требуется вывести одно число – длину самой продолжительной оттепели, то есть наибольшее количество последовательных дней, на протяжении которых среднесуточная температура превышала 0 градусов. Если температура в каждый из дней была неположительной, выведите 0.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 6<br>-20 30 -40 50 10 -10 | 2 |
| 2 | 8<br>10 20 30 1 -10 1 2 3 | 4 |
| 3 | 5<br>-10 0 -10 0 -10 | 0 |

## 265. Шахматная доска

*(Время: 1 сек. Память: 16 Мб Сложность: 36%)*

Из шахматной доски по границам клеток выпилили связную (не распадающуюся на части) фигуру без дыр. Требуется определить ее периметр.

**Входные данные**

Во входном файле INPUT.TXT сначала записано число N ($1 \le N \le 64$) – количество выпиленных клеток. В следующих N строках указаны координаты выпиленных клеток, разделенные пробелом (номер строки и столбца – числа от 1 до 8). Каждая выпиленная клетка указывается один раз.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите одно число – периметр выпиленной фигуры (сторона клетки равна единице).

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3<br>1 1<br>1 2<br>2 1 | 2 |
| 2 | 1<br>8 8 | 4 |

## 266. Кассы

*(Время: 1 сек. Память: 16 Мб Сложность: 39%)*

На одном из московских вокзалов билеты продают N касс. Каждая касса работает без перерыва определенный промежуток времени по фиксированному расписанию (одному и тому же каждый день). Требуется определить, на протяжении какого времени в течение суток работают все кассы одновременно.

**Входные данные**

Во входном файле INPUT.TXT сначала располагается одно целое число N ($0 < N \le 1000$). В каждой из следующих N строк через пробел расположены 4 целых числа, первые два из которых обозначают время открытия кассы в часах и минутах (часы – целое число от 0 до 23, минуты – целое число от 0 до 59), остальные два – время закрытия в том же формате. Числа разделены пробелами.

Время открытия означает, что в соответствующую ему минуту касса уже работает, а время закрытия – что в соответствующую минуту касса уже не работает. Например, касса, открытая с 10 ч 30 мин до 18 ч 30 мин, ежесуточно работает 480 минут.

Если времена открытия и закрытия совпадают, то это означает, что касса работает круглосуточно. Если первое время больше второго, то это означает, что касса начинает работу до полуночи, а заканчивает – на следующий день.

**Выходные данные**

В выходной файл OUTPUT.TXT требуется вывести одно число – суммарное время за сутки (в минутах), на протяжении которого работают все N касс.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3<br>1 0 23 0<br>12 0 12 0<br>22 0 2 0 | 120 |
| 2 | 2<br>9 30 14 0<br>14 15 21 0 | 0 |
| 3 | 2<br>14 00 18 00<br>10 00 14 01 | 1 |

## 267. Ксерокопии

*(Время: 1 сек. Память: 16 Мб Сложность: 38%)*

Секретарша Ирочка сегодня опоздала на работу и ей срочно нужно успеть к обеду сделать N копий одного документа. В ее распоряжении имеются два ксерокса, один из которых копирует лист за x секунд, а другой – за y секунд. Разрешается использовать как один ксерокс, так и оба одновременно. Можно копировать не только с оригинала, но и с копии. Помогите ей выяснить, какое минимальное время для этого потребуется.

**Входные данные**

Во входном файле INPUT.TXT записаны три натуральных числа N, x и y, разделенные пробелом ($1 \le N \le 2 \cdot 10^8$, $1 \le x, y \le 10$).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите одно число – минимальное время в секундах, необходимое для получения N копий.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 4 1 1 | 3 |
| 2 | 5 1 2 | 4 |

## 268. Почти палиндром

*(Время: 5 сек. Память: 64 Мб Сложность: 55%)*

Слово называется палиндромом, если его первая буква совпадает с последней, вторая – с предпоследней и т.д. Например: «abba», «madam», «x».

Для заданного числа K слово называется почти палиндромом, если в нем можно изменить не более K любых букв так, чтобы получился палиндром. Например, при K = 2 слова «reactor», «kolobok», «madam» являются почти палиндромами (подчеркнуты буквы, заменой которых можно получить палиндром).

Подсловом данного слова являются все слова, получающиеся путем вычеркивания из данного нескольких (возможно, одной или нуля) первых букв и нескольких последних. Например, подсловами слова «cat» являются слова «c», «a», «t», «ca», «at» и само слово «cat» (а «ct» подсловом слова «cat» не является).

Требуется для данного числа K определить, сколько подслов данного слова S являются почти палиндромами.

**Входные данные**

В первой строке входного файла INPUT.TXT вводятся два натуральных числа: N ($1 \le N \le 5000$) – длина слова и K ($0 \le K \le N$). Во второй строке записано слово S, состоящее из N строчных латинских букв.

**Выходные данные**

В выходной файл OUTPUT.TXT требуется вывести одно число – количество подслов слова S, являющихся почти палиндромами (для данного K).

**Примеры**

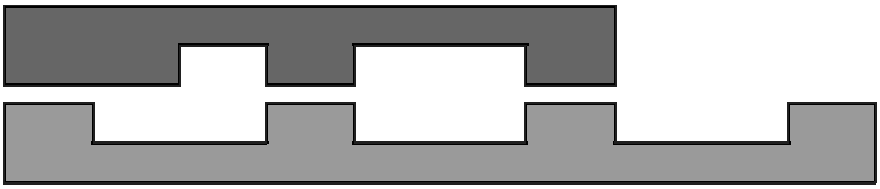
| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 5 1<br>abcde | 3 |
| 2 | 3 3<br>aaa | 4 |

## 269. Тормозной механизм

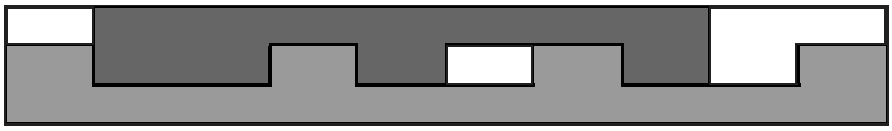
*(Время: 1 сек. Память: 16 Мб Сложность: 40%)*

Исследовательская лаборатория одной известной автомобильной компании разработала специальный механизм, позволяющий повысить эффективность тормозов путем равномерной нагрузки деталей, используемых в тормозах.

Одним из основных компонентов механизма являются 2 прокладки, которые в процессе взаимодействия накладываются друг на друга. Каждая прокладка длины n разделена на n разделов, каждый из которых имеет высоту h или 2h. Таким образом, прокладки имеют зубчатую форму без закруглений.



В процессе взаимодействия прокладок важно, чтобы они накладывались друг на друга и при этом общая длина получившегося соединения была наименьшей.



По заданной конфигурации прокладок требуется определить наименьшую длину их возможного соединения, при котором общая высота конструкции не превышает значения 3h. При этом вращать прокладки и удалять зубцы запрещено.

**Входные данные**

Входной файл INPUT.TXT содержит 2 строки с описанием конфигурации 2-х прокладок. Каждая конфигурация определяется последовательностью цифр 1 и 2, соответствующих высоте каждого зубца прокладки. Каждая из строк не пуста и имеет длину, не превышающую 100.

**Выходные данные**

В выходной файл OUTPUT.TXT требуется вывести наименьшую длину конструкции из заданных прокладок.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 2112112112<br>2212112 | 10 |
| 2 | 12121212<br>21212121 | 8 |
| 3 | 2211221122<br>21212 | 15 |

## 270. Java vs C++

*(Время: 1 сек. Память: 16 Мб Сложность: 44%)*

Сторонники языков Java и C++ часто спорят о том, какой язык лучше для решения олимпиадных задач. Одни говорят, что в Java есть масса полезных библиотек для работы со строками, хорошо реализованы механизмы чтения и вывода данных, а так же радует встроенные возможности для реализации длинной арифметики. С другой стороны, C++ является классическим языком, скорость выполнения программ благодаря существующим компиляторам (например, Intel Compiler 10.0) гораздо выше, чем у Java.

Но сейчас нас интересует лишь небольшие отличия, а именно соглашения, которыми пользуются программисты при описании имен переменных в Java и C++. Известно, что для понимания значений переменных часто используют английские слова или даже целые предложения, описывающие суть переменных, содержащих те или иные значения. Приведем ниже правила описания переменных, которыми руководствуются программисты, реализующие программы на Java и C++.

В языке Java принято первое слово, входящее в название переменной записывать с маленькой латинской буквы, следующее слово идет с большой буквы (только первая буква слова большая), слова не имеют разделителей и состоят только из латинских букв. Например, правильные записи переменных в Java могут выглядеть следующим образом: javaIdentifier, longAndMnemonicIdentifier, name, nEERC.

В языке C++ для описания переменных используются только маленькие латинские символы и символ «_», который отделяет непустые слова друг от друга. Примеры: java_identifier, long_and_mnemonic_identifier, name, n_e_e_r_c.

Вам требуется написать программу, которая преобразует переменную, записанную на одном языке в формат другого языка.

**Входные данные**

Во входном файле INPUT.TXT задано наименование переменной.

**Выходные данные**

В выходной файл OUTPUT.TXT требуется вывести аналог имени переменной в другом языке. Т.е. если переменная представлена в формате Java, то следует перевести в формат C++ и наоборот. В том случае, когда имя переменной не соответствует ни одному из вышеописанных языков, следует вывести «Error!»

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | `long_and_mnemonic_identifier` | `longAndMnemonicIdentifier` |
| 2 | `anotherExample` | `another_example` |
| 3 | `i` | `i` |
| 4 | `bad_Style` | `Error!` |

## 271. Число Фибоначчи

*(Время: 1 сек. Память: 16 Мб Сложность: 20%)*

Числа Фибоначчи строятся следующим образом: 1, 1, 2, 3, 5, …. В этой последовательности, начиная с третьего числа, каждый следующий член равен сумме двух предыдущих. Получаем, что, например, шестое число равно 8, а десятое - 55.

Требуется написать программу, которая определяет, является ли заданное число числом Фибоначчи.

**Входные данные**

Входной текстовый файл INPUT.TXT содержит одно натуральное число в диапазоне от 2 до 1200000000.

**Выходные данные**

Выходной файл OUTPUT.TXT должен содержать в первой строке 1, если заданное число является числом Фибоначчи, и 0, иначе. В первом случае во вторую строку требуется вывести его порядковый номер.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 8 | 1<br>6 |
| 2 | 10 | 0 |

## 272. Сумма максимума и минимума

*(Время: 1 сек. Память: 16 Мб Сложность: 26%)*

Задана последовательность целых чисел. Числа нумеруются по порядку следования, начиная с единицы.

Требуется написать программу, которая найдет сумму максимума из чисел с четными номерами и минимума из чисел с нечетными номерами – $\max\{a_2, a_4, \ldots\}+\min\{a_1, a_3, \ldots\}$.

**Входные данные**

Входной текстовый файл INPUT.TXT содержит в единственной строке последовательность от 2 до $10^5$ целых чисел, которые по модулю не превышают 10000.

**Выходные данные**

Выходной текстовый файл OUTPUT.TXT должен содержать одно целое число – сумму максимума из чисел с четными номерами и минимума из чисел с нечетными номерами.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 1 2 | 3 |
| 2 | 1 -2 3 -4 5 | -1 |

## 273. Вычеркивание

*(Время: 1 сек. Память: 16 Мб Сложность: 28%)*

Задано натуральное число N. Требуется написать программу, вычисляющую количество различных трехзначных чисел получающихся из N вычеркиванием цифр из его десятичной записи.

**Входные данные**

Входной текстовый файл INPUT.TXT содержит одно натуральное число N ($1 \le N \le 10^{100}$).

**Выходные данные**

Выходной текстовый файл OUTPUT.TXT должен содержать одно натуральное число – найденное количество трехзначных чисел.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 12 | 0 |
| 2 | 11111111110011111111 | 4 |

## 274. Дружные числа

*(Время: 1 сек. Память: 16 Мб Сложность: 25%)*

Будем называть два числа дружными, если они состоят из одних и тех же цифр. Например, числа 1132 и 32321 являются дружными, а 12 и 123 – нет (в первом числе нет цифры 3). Требуется написать программу, которая определит, являются ли два заданных числа дружными.

**Входные данные**

Входной текстовый файл INPUT.TXT содержит в первой строке натуральное число K – количество тестов. Количество тестов не превышает 10. В следующих K строках содержатся по два целых числа A и B, разделенные одним пробелом ($0 < A < 10^9$, $0 < B < 10^9$).

**Выходные данные**

Выходной текстовый файл OUTPUT.TXT должен содержать K строк. Для каждого теста в отдельной строке надо выдать сообщение “YES”, если A и B являются дружными, или “NO”, если не являются. В сообщениях кавычки не печатать.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 1<br>1132 32321 | YES |
| 2 | 2<br>12 123<br>11 111 | NO<br>YES |

## 275. Делимость на 7

*(Время: 1 сек. Память: 16 Мб Сложность: 42%)*

Требуется определить делимость на 7 ряда целых чисел, записанных в двоичной системе счисления.

**Входные данные**

В первой строке входного файла INPUT.TXT содержится N – количество чисел (N < 50). В следующих N строках содержатся двоичные числа (по одному в каждой строке). Каждое двоичное число состоит не более чем из 1000 цифр.

**Выходные данные**

Выходной файл OUTPUT.TXT должен содержать N строк. Для каждого теста в отдельной строке надо выдать сообщение «Yes», если соответствующее число кратно 7 или «No» в противном случае.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3<br>1110<br>1010101<br>11111111111111111111111111 | Yes<br>No<br>Yes |
| 2 | 1<br>11 | No |

**Разбор**

На первый взгляд, задача может показаться достаточно сложной в реализации и многие, возможно, прибегнут к тяжелому решению: переводу длинного двоичного числа в десятичное и многократному делению длинного десятичного на 7; либо к делению длинного двоичного на двоичное 7. В любом из этих случаев не избежать достаточно сложных механизмов реализации длинной арифметики. Причем в данных ограничениях неэффективная реализация может привести к превышению времени выполнения программы.

Используем следующее свойство: число в системе счисления с основанием K делится на K-1 тогда и только тогда, когда сумма цифр данного числа так же делится на K-1. Всем известно, что в десятичной системе число делится на 9 тогда и только тогда, когда сумма его цифр делится на 9. А вышеописанное свойство – это некоторое обобщение известного всем факта. В нашем случае мы можем это использовать. Для этого следует перевести заданное число в 8-ную систему счисления, сложить полученные цифры и проверить делимость на 7 уже обычного числа. Напомним, что каждая тройка цифр двоичного числа представляет отдельную цифру того же числа в 8-ой системе счисления.

Алгоритм проверки делимости двоичного числа S на 7 можно описать следующим образом:

```
read(S);
while(len(S) mod 3 > 0) S = '0'+S;
while(s>''){
  x = x+s[1]*4+s[2]*2+s[3];
  delete(s,1,3);
}
if(x mod 7 = 0) write('Yes') else write('No');
```

Используя представленный выше алгоритм, вам не составит труда решить данную задачу, где напомним, проверить на делимость нужно несколько чисел.

## 276. Разбиение на части

*(Время: 1 сек. Память: 16 Мб Сложность: 21%)*

Необходимо представить целое число N в виде суммы M примерно равных чисел. Будем считать, что числа примерно равны, если они отличаются друг от друга не более чем на единицу.

**Входные данные**

Во входном файле INPUT.TXT записаны два натуральных числа N и M через пробел, каждое из которых не превосходит 30000.

**Выходные данные**

Выходной файл OUTPUT.TXT должен содержать M примерно равных чисел, сумма которых должна быть равна N. Все числа следует вывести в одной строке в порядке неубывания через пробел.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 13 4 | 3 3 3 4 |
| 2 | 72 8 | 9 9 9 9 9 9 9 9 |

## 277. Школьная алгебра

*(Время: 1 сек. Память: 16 Мб Сложность: 27%)*

Трёхчлен a + bx + cy от двух переменных x и y однозначно определяется коэффициентами a, b и c. Написать программу, которая по заданным a, b и c выводит соответствующий трёхчлен, записанный с использованием алгебраических соглашений:

- коэффициент при члене, содержащем переменную, опускается, если его модуль равен единице;
- член, коэффициент при котором равен нулю, опускается (кроме случая, когда все коэффициенты равны нулю, тогда трехчлен состоит из одной цифры 0);
- знак «+» опускается, если он предшествует отрицательному коэффициенту;
- знак «+» опускается, если он стоит в начале выражения (так называемый унарный плюс);
- знак умножения между коэффициентом и переменной опускается.

При этом запрещено менять местами члены.

**Входные данные**

Во входном файле INPUT.TXT через пробел записаны целые коэффициенты a, b и c, каждое из которых не превосходит 30000 по абсолютной величине.

**Выходные данные**

Выходной файл OUTPUT.TXT должен содержать трехчлен, записанный с использованием алгебраических соглашений.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 0 2 -1 | 2x-y |
| 2 | 3 0 -2 | 3-2y |

## 278. Вычислительная биология

*(Время: 1 сек. Память: 16 Мб Сложность: 28%)*

В современной биологии ученым часто приходится иметь дело с последовательностями ДНК. Эти последовательности зачастую являются очень длинными, и их ручная обработка требует большого количества времени и сил. Поэтому возникает идея автоматизировать этот процесс.

Для этого можно применять компьютерные методы обработки данных, например, весьма полезными оказываются алгоритмы на строках. В этой задаче последовательность ДНК будет представляться в виде строки, все символы которой входят в множество {A, G, C, T}.

Пусть даны две последовательности ДНК: $s = s_1 s_2 \ldots s_n$ и $t = t_1 t_2 \ldots t_m$. Будем говорить, что t может получится в результате эволюции из s, если s является подпоследовательностью t, то есть существует такая последовательность индексов $1 < i_1 < i_2 < \ldots i_m$, что $s_1=t_{i1}$, $s_2=t_{i2}$, $\ldots$ $s_m=t_{im}$. Необходимо выяснить, может ли последовательность t получиться в результате эволюции из s.

**Входные данные**

Первая строка входного файла INPUT.TXT содержит последовательность s, вторая – последовательность t. Размер входного файла не превосходит 256 килобайт.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите слово YES, если последовательность t могла получиться в результате эволюции из s, и слово NO – иначе.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | GTA<br>AGCTA | YES |
| 2 | AAAG<br>GAAAAAT | NO |

## 279. Скобочки - 2

*(Время: 1 сек. Память: 16 Мб Сложность: 57%)*

Напомним, что называется правильной скобочной последовательностью:

- пустая строка является правильной скобочной последовательностью;
- если строка a – правильная скобочная последовательность, то строки (a), [a] – тоже правильные скобочные последовательности;
- если строки a и b – правильные скобочные последовательности, то строка ab – тоже правильная скобочная последовательность.

Задана строка S, состоящая из квадратных и круглых скобок. Разрешается заменять квадратную открывающую скобку ([) на круглую открывающую (() и наоборот, а также квадратную закрывающую скобку (]) на круглую закрывающую ()) и наоборот.

За одно действие разрешается изменить ровно один символ строки. Необходимо за минимальное число действий преобразовать S в правильную скобочную последовательность.

**Входные данные**

Входной файл INPUT.TXT содержит строку S. Ее длина не превосходит 100 000 символов.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите искомое минимальное число действий или -1, если преобразовать S в правильную скобочную последовательность невозможно.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | ())[] | 0 |
| 2 | [(]) | 2 |
| 3 | ((]] | -1 |

## 280. Количество делителей - 2

*(Время: 1 сек. Память: 16 Мб Сложность: 51%)*

Пусть x – натуральное число. Назовем y его делителем, если $1 \le y \le x$ и остаток от деления x на y равен нулю.

Задано число x. Найдите количество его делителей, делящихся на каждое из простых чисел, на которое делится x.

**Входные данные**

Входной файл INPUT.TXT содержит целое число x ($1 \le x \le 10^{18}$). Все простые делители числа x не превосходят тысячу.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите ответ на задачу.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 12 | 2 |
| 2 | 239 | 1 |

## 281. Игра с монеткой

*(Время: 1 сек. Память: 16 Мб Сложность: 39%)*

Петя играет в интересную игру. Для этой игры необходима монетка. Петя подбрасывает ее n раз и считает, сколько раз выпадает «решка». Если решка выпадает хотя бы m раз, то Петя считает, что он выиграл игру.

Однажды Петя задумался, какова вероятность того, что он выиграет игру. Для этого он хочет найти количество последовательностей результатов подбрасывания монетки, содержащих ровно n подбрасываний, при которых «решка» выпала хотя бы m раз.

Помогите Пете – найдите это число, считая, что при каждом броске монетка может выпасть либо «орлом», либо «решкой».

**Входные данные**

Входной файл INPUT.TXT содержит два целых числа: n и m ($1 < n < 20$, $0 < m < n$).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите ответ на задачу.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 2 0 | 4 |
| 2 | 3 2 | 4 |

## 282. Прямоугольники

*(Время: 1 сек. Память: 16 Мб Сложность: 47%)*

Найдите количество невырожденных прямоугольников со сторонами, параллельными осям координат, вершины которых лежат в точках с целыми координатами внутри или на границе прямоугольника, противоположные углы которого находятся в точках (0, 0) и (W, H).

**Входные данные**

Входной файл INPUT.TXT содержит два натуральных числа W и H, не превосходящих 1000.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите ответ на задачу.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 1 1 | 1 |
| 2 | 2 1 | 3 |

## 283. Рунные слова

*(Время: 1 сек. Память: 16 Мб Сложность: 25%)*

Руны – это древние магические знаки, которые наши предки использовали как буквы. Говорят, что рунные знаки обладают магическими свойствами, а при сложении рун в слова их магическая сила многократно возрастает. Если кузнец изготовит доспехи и начертит там определенные руны в определенном порядке, то доспехи будут наделены необычайными магическими силами.

Для того, чтобы стать обладателем таких доспехов достаточно просто принести кузнецу начертания этих рунных знаков. А вот, чтобы стать обладателем рунного знака приходилось немало потрудиться. Воины добывали начертания рун других языков и наречий в боях или получали их в качестве наград в благодарность за оказанные услуги.

Но так или иначе и в этом деле развелись жулики. По подозрениям ученых кузнец Игнатус Мошеникус изготавливал благородным войнам фальшивые рунные слова. Из древних преданий ученым стало достоверно известно, что каждая руна записывается из двух, трех или четырех латинских букв. Причем первая буква рунного слова всегда записывается как заглавная, а все остальные являются маленькими. Ученые перевели несколько, выкованных этим кузнецом, рунных слов на латинский язык и теперь нуждаются в Вашей помощи. Проверьте, является ли приведенное слово рунным.

**Входные данные**

В единственной строке входного файла INPUT.TXT содержится слово. Оно представляет собой непустую строку, длиной не более 100000 символов, содержащую только большие и маленькие буквы латинского алфавита.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите «Yes», если слово является рунным и «No» в противном случае.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | IoIsTheBest | Yes |
| 2 | IoItIsWaste | No |

## 284. Подмассив массива

*(Время: 1 сек. Память: 16 Мб Сложность: 15%)*

Пусть задан массив целых чисел $a_1$, $a_2$, ..., $a_n$. Назовем его подмассивом f(i,j) массив, составленный из чисел массива $a_i$, $a_{j+1}$,..., $a_{j-1}$, $a_j$. Напишите программу, которая будет выводить подмассивы массива a.

**Входные данные**

Первая строка входного файла INPUT.TXT содержит число n ($1 \le n \le 1000$) - количество элементов в массиве a. Во второй строке содержатся числа $a_1$, $a_2$, … , $a_n$ разделенные пробелом. Все $a_i$ находятся в диапазоне от $-2^{31}$ до $2^{31} - 1$. В третьей строке находится m ($1 \le m \le 100$) – количество подмассивов, которые необходимо вывести. Следующие m строк содержат пары чисел $i_k$, $j_k$ ($1 \le i_k \le j_k \le n$).

**Выходные данные**

В выходной файл OUTPUT.TXT для каждой пары ($i_k$,$j_k$) в отдельной строке выведите подмассив f($i_k$,$j_k$).

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 6<br>1 2 3 4 5 6<br>5<br>1 1<br>2 6<br>3 4<br>5 6<br>2 4 | 1<br>2 3 4 5 6<br>3 4<br>5 6<br>2 3 4 |

## 285. Костер

*(Время: 1 сек. Память: 16 Мб Сложность: 34%)*

Во время военного похода на морского пехотинца Джо было возложено ответственное задание – развести костёр и поддерживать в нём огонь ровно m минут. Для этого у Джо есть спички и n поленьев, причём Джо известно точное время сгорания каждого полена.

Джо разжигает огонь в момент времени t = 0 и сразу бросает в него одно или несколько поленьев. Затем он должен подбрасывать в огонь новые поленья, не позволяя костру угаснуть (т.е. если последнее полено в костре догорает в момент времени t, то новое полено может быть брошено в огонь не позднее t – 1). Поленья, брошенные в огонь, загораются мгновенно. Одновременно Джо может бросить в огонь любое количество поленьев. Джо должен бросить в огонь все n поленьев.

Помогите Джо определить, сможет ли он подбрасывать поленья в огонь таким образом, чтобы костер горел ровно m минут.

**Входные данные**

Первая строка входного файла INPUT.TXT содержит целые числа n и m ($1 \le n \le 100$, $1 \le m \le 1000$) – количество поленьев и время, в течение которого Джо должен поддерживать огонь в костре. Вторая строка входного файла содержит n целых чисел $a_1$, $a_2$, ..., $a_n$, где $a_i$ ($2 \le a_i \le 1000$) – время сгорания i-ого полена в минутах.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите строку «yes», если Джо сможет поддерживать огонь в костре ровно m минут, и строку «no» в противном случае.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3 7<br>2 3 5 | yes |
| 2 | 2 5<br>3 9 | no |
| 3 | 4 10<br>3 3 3 3 | no |

## 286. Больше-меньше - 2

*(Время: 1 сек. Память: 16 Мб Сложность: 30%)*

Программист Билл занимается разработкой программного обеспечения для новейшего робота-исследователя, которого учёные планируют отправить на Марс с целью поиска там следов разумной жизни. Модули, которые отвечают за передвижение робота и сбор проб грунта, Билл уже скачал из Интернета. Оставалось лишь научить робота отличать разумные формы жизни от неразумных. Для этого Боб несколько месяцев посещал программистские форумы, и, наконец, нашёл подходящий модуль. Теперь, чтобы определить, является ли тот или иной объект представителем внеземной расы, роботу достаточно сравнить два вещественных числа.

Однако за несколько часов до запуска корабля на Марс обнаружилось, что робот неправильно сравнивает вещественные числа! Чтобы исправить эту ошибку, учёные обратились за помощью к Вам.

**Входные данные**

Входной файл INPUT.TXT состоит из двух строк, в каждой из которых записано по одному вещественному числу без ведущих нулей. Целая и дробная части отделяются точкой, которая может быть опущена, если число целое. Каждое из чисел содержит не более 10000 цифр.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите один символ «<», если первое число меньше второго, «>», если больше, и «=», если числа равны.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 2.39<br>3.61 | < |
| 2 | 123<br>12.3 | > |

| | | |
|---|---|---|
| 3 | 12345678<br>12345678.0 | = |
| 4 | -1.0<br>1.0 | < |

## 287. Профессор

*(Время: 1 сек. Память: 16 Мб Сложность: 36%)*

В одном очень известном университете один очень известный профессор очень быстро произносил свои лекции, так, что ничего невозможно было разобрать. Но недавно студент Вилли решил провести исследование по изучению словарного запаса профессора. С этой целью он даже посетил одну лекцию и записал всё сказанное на ней на диктофон. Затем, прокручивая дома запись с десятикратным замедлением, Вилли смог записать всё, что сказал профессор.

Но вот незадача – профессор говорил так быстро, что, даже прослушивая замедленную запись, нельзя было точно сказать, где он делал паузы между словами. Таким образом, у Вилли есть некоторый текст, состоящий из n маленьких латинских букв – лекция, которая была прочитана профессором. Теперь Вилли хочет знать, какое количество различных слов длины m мог использовать в своей лекции профессор.

**Входные данные**

Первая строка входного файла INPUT.TXT содержит два числа n и m ($1 \le m \le n \le 100$) – длина лекции и длина слова.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите одно число – количество слов длины m, которые профессор мог использовать в своей лекции.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3 1<br>abc | 3 |
| 2 | 10 3<br>bbaabbbabb | 6 |

## 288. Комментарии

*(Время: 1 сек. Память: 16 Мб Сложность: 52%)*

Программист Билл недавно узнал, что, чем больше комментариев содержит исходный текст, тем он лучше. Теперь он хочет проверить, насколько хороши его собственные программы, написанные на языке Pascal. Но поскольку самому считать комментарии очень утомительно, Билл попросил Вас сделать эту работу за него.

Исходный текст может содержать комментарии трёх типов:

1. // ...
2. { ... }
3. (* ... *)

Комментарий первого типа начинается составным символом // и продолжается до конца строки. Комментарий второго типа начинается символом { и заканчивается символом }. Он может размещаться в нескольких строках. Комментарий третьего типа начинается составным символом (* и заканчивается составным символом *). Он также может размещаться в нескольких строках.

Комментарии не могут быть вложены друг в друга, так что запись вида {…//…(*…*)…} является одним комментарием второго типа. Комментарии не могут размещаться внутри символьных строк, так что запись ‘…(**)…{}…’ не содержит ни одного комментария.

**Входные данные**

Во входном файле INPUT.TXT записан исходный текст программы на языке Pascal. Размер текста не превосходит 16 Кб.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите одно число – количество комментариев в исходном тексте программы.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | `{ my first program }`<br>`begin`<br>`writeln('hello world');`<br>`end.` | 1 |
| 2 | `const comments = '{}(**)//';`<br>`begin`<br>`writeln(comments);`<br>`end.` | 0 |
| 3 | `// comment`<br>`Begin`<br>`writeln('{ string }');`<br>`{ while (true) do; }`<br>`end.` | 2 |

## 289. Делители

*(Время: 1 сек. Память: 16 Мб Сложность: 76%)*

По заданному количеству делителей числа требуется найти само это число.

**Входные данные**

Во входном файле INPUT.TXT записано количество делителей D некоторого числа N ($1 \le D \le 5000$).

**Выходные данные**

В выходной файл OUTPUT.TXT запишите число N. Если решений несколько, выведите наименьшее из них. Если решения нет, или наименьшее из решений превосходит $10^{15}+1$, запишите в файл число 0.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3 | 4 |
| 2 | 4 | 6 |
| 3 | 12 | 60 |
| 4 | 60 | 5040 |
| 5 | 4911 | 0 |

## 290. База террористов

*(Время: 1 сек. Память: 16 Мб Сложность: 42%)*

Спецслужбы получили информацию о том, что в труднодоступной части Муравийской пустыни расположена хорошо замаскированная база террористов. В руки спецслужб попал и план этой базы, которая с большой высоты выглядит как группа скальных обломков, весьма часто встречающихся среди Муравийских песков. Президент отдал приказ уничтожить базу крылатыми ракетами. Ваша задача по карте пустыни, полученной со спутника и плану базы определить количество возможных положений базы террористов. Помните, что террористы могли привезти на территорию базы камни!

**Входные данные**

В первой строке файла INPUT.TXT записаны числа $N_b$ и $M_b$ ($1 \le N_b, M_b \le 20$). В следующих $N_b$ строках записан план базы. Каждая из этих строк содержит по $M_b$ символов «#» (ASII 35 ) или «.» (ASCII 46). Символ «#» обозначает фрагмент базы, а символ «.» – песок. В

следующей строке записаны числа $N_d$ и $M_d$ ($1 \le N_d, M_d \le 100$). И остаток файла содержит карту участка пустыни на котором, предположительно, находится база террористов – $N_d$ строк по $M_d$ символов «#» или «.» в каждой.

**Выходные данные**

Запишите в файл OUTPUT.TXT количество возможных положений базы террористов.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 2 2<br>#.<br>##<br>3 5<br>#.#.#<br>#####<br>.###. | 4 |
| 2 | 1 3<br>#..<br>3 6<br>##..##<br>.#.#.#<br>#.#... | 6 |
| 3 | 3 3<br>#..<br>#.#<br>#..<br>5 36<br>#......#.....#...<br>#.#....#.#...#.....#.....<br>#......#.#...##....#...##.....###..<br>.......#.....#...#.#...##.....#.#<br>.....#.....#.....#..... | 4 |

## 291. Словарь

*(Время: 1 сек. Память: 16 Мб Сложность: 31%)*

Дан некоторый набор букв и словарь. Ваша задача – подсчитать, сколько различных слов из словаря можно составить из этих букв.

**Входные данные**

В первой строке файла INPUT.TXT записано число N – количество слов в словаре ($0 \le N \le 1000$). В следующих N строках файла записано по одному слову из словаря. Слова содержат от 1 до 10 маленьких латинских букв. Все слова в словаре различны. В последней строке файла записан набор букв (не более 100 букв).

**Выходные данные**

Запишите в файл OUTPUT.TXT количество различных слов из словаря, которые можно составить из заданного набора букв.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 7<br>ant<br>bee<br>cat<br>dog<br>ewe<br>fly<br>gnu<br>bew | 0 |

| | | |
|---|---|---|
| 2 | 7<br>bee<br>fly<br>cat<br>dog<br>ant<br>ewe<br>gnu<br>tancugd | 3 |

## 292. Простой цифровой корень

*(Время: 1 сек. Память: 16 Мб Сложность: 36%)*

Определим простой цифровой корень (ПЦК) натурального числа N следующим образом. Если N – простое число, то ПЦК(N) = N. Если число однозначное, но не простое (то есть 1, 4, 6, 8 или 9), то ПЦК(N) = 0. В остальных случаях ПЦК(N) = ПЦК(S(N)), где S(N) – сумма цифр числа N.

**Входные данные**

Во входном файле INPUT.TXT записано число N ($1 \le N \le 2^{31}-1$).

**Выходные данные**

Запишите в файл OUTPUT.TXT простой цифровой корень числа N.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 1 | 0 |
| 2 | 3 | 3 |
| 3 | 128 | 11 |

## 293. Налоги

*(Время: 1 сек. Память: 16 Мб Сложность: 20%)*

В некотором государстве действует N фирм, конкурирующих между собой. У каждой фирмы есть некоторая прибыль в год, равная V[i] американских рублей. У царя есть любимые фирмы, а есть нелюбимые. Соответственно, налог для всех фирм разный и назначается царем в индивидуальном порядке. Налог на i-ую фирму равен p[i] процентов.

Собиратели статистики решили посчитать, с какой фирмы в государственную казну идет наибольший доход (в казну идут все налоги). К сожалению, они не учили в детстве ни математику, ни информатику (так что учитесь, дети!), и их задача резко осложняется.

Помогите им в этой нелегкой задаче.

**Входные данные**

Во входном файле INPUT.TXT сначала записано число N – число фирм ($0 < N \le 100$). Далее идет N целых неотрицательных чисел, не превышающих 154 – доходы фирм, а затем еще N целых чисел от 0 до 100 – налоги фирм в процентах.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите одно число – номер фирмы, от которой государство получает наибольший налог. Если таких фирм несколько, выведите фирму с наименьшим номером.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 1<br>1<br>1 | 1 |

| | | |
|---|---|---|
| 2 | 2<br>1 2<br>3 2 | 2 |
| 3 | 3<br>100 1 50<br>0 100 3 | 3 |

## 294. Болты и гайки

*(Время: 1 сек. Память: 16 Мб Сложность: 17%)*

Вновь созданная фирма купила заброшенные склады на окраине города. Новому заведующему складами поручили произвести учёт в короткие сроки. Всё шло хорошо, пока случайно не рассыпали контейнеры с болтами и гайками на каждом складе, после чего собрали их в общие (для болтов и гаек) контейнеры, потеряв при этом несколько деталей.

Помогите оценить нанесённый ущерб на каждом складе, приняв во внимание, что, помимо потерянных деталей, болт (или гайка) считается непригодным, если он не имеет соответствующей гайки (или болта).

**Входные данные**

Во входном файле INPUT.TXT описано текущее положение на складе. В первой строке через пробел записаны три целых числа: k1, l1, m1 – начальное число болтов ($100 \le k1 \le 30000$, k1 кратно 100), процент потерянных деталей ($0 \le l1 \le 100$) и стоимость одного болта ($1 \le m1 \le 100$) соответственно. Во второй строке через пробел записаны также три целых числа: k2, l2, m2 – начальное число гаек ($100 \le k2 \le 30000$, k2 кратно 100), процент потерянных деталей ($0 \le l2 \le 100$) и стоимость одной гайки ($1 \le m2 \le 100$) соответственно.

**Выходные данные**

В выходной OUTPUT.TXT выведите одно целое число – размер ущерба.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 1000 10 100<br>1200 20 90 | 37000 |
| 2 | 5000 15 23<br>4000 17 22 | 53600 |

## 295. Шифровка

*(Время: 1 сек. Память: 16 Мб Сложность: 29%)*

Разведкой был перехвачен ряд шифровок, которые передавал Джеймс Бонд. Известно, что каждое послание зашифровано методом циклического сдвига. Суть которого в том, что каждая буква заменяется на букву, отстоящую в алфавите от первой на определенном расстояние. Это расстояние называется знаменателем шифра. Так, при знаменателе шифра 2 буква D превратится в F, буква Q – в S, а Z – в B. Известно, что Бонд использует знаменатели от 0 до 25, и составляет послания исключительно из заглавных букв английского алфавита. Знаменатели в шифровках постоянно меняются, так что расшифровать содержимое послания будет не просто. После тщательного анализа удалось примерно определить предмет посланий. Теперь для каждого послания точно известно одно из входящих туда слов.

**Входные данные**

В первой строке входного файла INPUT.TXT содержится строка с перехваченным посланием, а во второй строке – слово, которое обязательно присутствует в этом послании. Обе строки состоят только из заглавных английских букв и содержат не больше 40 символов.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите расшифрованный текст, либо сообщение «IMPOSSIBLE», если разгадать шифровку невозможно. В тех случаях, когда расшифровка возможна с различными знаменателями, то следует вывести вариант с наименьшим таким значением.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | HELLOAMERICA<br>AMERICA | HELLOAMERICA |
| 2 | KHOORDPHULFD<br>HELLOAMERICA | HELLOAMERICA |
| 3 | KHOORDPHULFD<br>KHOORDPHULFC | IMPOSSIBLE |

## 296. Лиса Алиса и кот Базилио

*(Время: 1 сек. Память: 16 Мб Сложность: 22%)*

Лиса Алиса и кот Базилио вырастили денежное дерево. И выросли на нем трехрублевые и пятирублевые золотые монеты. Лиса Алиса себе взяла трехрублевые монеты, а коту Базилио отдала пятирублевые монеты. Посетовав на свою скромность, она предложила впредь рассчитываться за покупки вместе, деньги давать без сдачи и минимальным числом монет. Известно, что они сделали покупку стоимостью N рублей, при этом они рассчитались без сдачи.

Вам следует написать программу, которая определяет: сколько монет внес кот Базилио, и сколько монет внесла лиса Алиса.

**Входные данные**

Во входном файле INPUT.TXT записано одно натуральное число N – стоимость покупки в рублях ($7 < N < 1000$).

**Выходные данные**

В выходной OUTPUT.TXT выведите два целых числа через пробел: число монет, которые отдал кот Базилио и число монет, которые отдала лиса Алиса.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 8 | 1 1 |
| 2 | 11 | 1 2 |
| 3 | 15 | 3 0 |

## 297. Кругляши

*(Время: 1 сек. Память: 16 Мб Сложность: 16%)*

Однажды в просторах рунета появился следующий ребус:

157892 = 3
203516 = 2
409578 = 4
236271 = ?

Никто так и не смог его разгадать. Позже оказалось, что число в правом столбце равно сумме «кругляшей», которые есть в цифрах числа, расположенного слева. Ваша задача написать программу, которая определяет, сколько кругляшей в числе.

**Входные данные**

Во входном файле INPUT.TXT записано целое число N ($0 \le N \le 10^{100}$).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите одно число – количество кругляшей в числе N.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 157892 | 3 |
| 2 | 203516 | 2 |
| 3 | 409578 | 4 |
| 4 | 236271 | 1 |

## 298. Стрелок

*(Время: 1 сек. Память: 16 Мб Сложность: 28%)*

Стрелок стоит в центре стрельбища. На стрельбище несколько мишеней. Пули стрелка пробивают мишени насквозь, не теряя скорости, и могут поразить все мишени, стоящие на одной линии.

Будем считать, что стрелок стоит в центре начала координат. Известны координаты всех мишеней (для простоты будем считать их геометрические размеры пренебрежимо малыми). Определите минимальное число выстрелов, необходимых стрелку для поражения всех мишеней.

**Входные данные**

Первая строка входного файла INPUT.TXT содержит натуральное число N – количество мишеней ($N \le 20$). Далее идет N строк с информацией о координатах каждой мишени, при этом в каждой строке указывается два целых числа через пробел X и Y ($-10 \le X, Y \le 10$).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите одно целое число – наименьшее количество выстрелов, необходимых для поражения всех мишеней.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 4<br>2 2<br>-2 2<br>-2 -2<br>2 -2 | 4 |
| 2 | 6<br>2 2<br>-2 2<br>-2 -2<br>2 -2<br>1 1<br>-1 3 | 5 |

## 299. Волейбол

*(Время: 1 сек. Память: 16 Мб Сложность: 56%)*

Партия в волейболе выигрывается командой, которая первой набирает 25 очков с преимуществом минимум в два очка. В случае равного счета 24-24, игра продолжается до достижения преимущества в 2 очка (26-24; 27-25).

Две сыгранные партии, закончившиеся с одинаковым счетом, будем считать разными, если строки, в которых выписан порядок набора очков командами, не равны.

Комитет по проведению соревнований по волейболу заинтересовался количеством различных партий, заканчивающихся счетом 25:23. Их оказалось 16123801841550.

Определить, сколько существует различных партий, заканчивающихся заданным счетом.

**Входные данные**

Во входном файле INPUT.TXT указан конечный счет в партии (то есть такой, при котором победа в партии отдаётся одной из команд). Также известно, что ни одна из команд не набрала более 40 очков.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите количество всевозможных партий, которые заканчиваются данным счетом.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 25:12 | 1251677700 |

| | | |
|---|---|---|
| 2 | 20:25 | 1761039350070 |
| 3 | 25:23 | 16123801841550 |

## 300. Радар

*(Время: 1 сек. Память: 16 Мб Сложность: 38%)*

Радар подвергается атаке из четырех точек, являющихся вершинами квадрата, в центре которого и стоит радар. Радар укомплектован специальным щитом, позволяющим блокировать удар, но щит может защищать радар только с одной из четырех сторон, и поворот щита требует времени. Изначально щит направлен в сторону той вершины, откуда будет первая атака. Известно время запуска и скорость ракет, ведущих атаку.

Требуется определить, сколько ракет удастся отбить.

**Входные данные**

Первые четыре строки входного файла INPUT.TXT содержат время запуска в секундах Tx ($0 < Tx \le 1000$) и скорость полета в метрах в секундуVx x-ой ракеты ($0 < Vx \le 1000$). Ракеты перечисляются по часовой стрелке Далее задано время в секундах, необходимое для поворота щита на 90 градусов Tpov ($0 < Tpov \le 1000$) и половина диагонали квадрата D – расстояние в метрах, предстоящее пролететь каждой из ракет ($0 < D < 1000$). Все числа – целые.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите «ALIVE», если радар уцелеет при всех выстрелах, в противном случае следует вывести число успешно отраженных ракет.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 0 10<br>5 10<br>10 10<br>15 10<br>5 100 | ALIVE |
| 2 | 0 10<br>10 10<br>5 10<br>15 10<br>5 100 | 1 |

## 301. Код

*(Время: 1 сек. Память: 16 Мб Сложность: 46%)*

В наши дни в космосе находятся сотни спутников, и все они обмениваются данными. При этом система распознавания сигналов работает по схеме «Свой-Чужой». Один из спутников отправляет запрос другому спутнику в формате двух целых чисел, а второй спутник отвечает первому так же двумя целыми числами. Первые два числа первого спутника представляют собой сумму цифр и количество цифр тех двух чисел, которыми должен ответить второй спутник. При этом в качестве ответа должны получиться числа, представляющие наибольшее и наименьшее возможные значения, которые могут быть сформированы по описанному выше методу.

Вам предстоит написать программу, формирующую ответ для второго спутника по известным числам, полученным от первого спутника.

**Входные данные**

Во входном файле INPUT.TXT записаны 2 натуральных числа S и K, представляющих сумму и количество цифр соответственно ($K \le 100$). При этом гарантируется, что возможно составить хотя бы одно K-значное число, сумма цифр которого равна S.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите два числа – ответ второго спутника. При этом следует помнить, что все числа не имеют лидирующих нулей.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 1 3 | 100 100 |
| 2 | 2 3 | 200 101 |
| 3 | 3 4 | 3000 1002 |

## 302. Города

*(Время: 5 сек. Память: 16 Мб Сложность: 65%)*

Для исследования поверхности Марса ученым необходимо разработать систему оповещения, которая смогла бы передавать информацию по цепному принципу между городами, которые планируется там построить.

При этом в каждом городе необходимо построить радиостанцию таким образом, чтобы была связь между всеми городами. При этом все такие станции должны передавать сигнал друг другу на равном расстоянии R. Таким образом, будет возможна передачи информации из одного города в другой только тогда, когда расстояние между ними не более R.

По заданным координатам городов, в целях экономии энергии радиостанций, Вам следует определить минимальное значение R, при котором информация сможет быть доставлена из любого города во все остальные.

**Входные данные**

В первой строке входного файла INPUT.TXT сначала записано натуральное число N – количество городов ($N \le 1000$). Далее идет N строк, содержащих вещественные координаты (Xi, Yi) соответствующего города ($-10000 \le Xi, Yi \le 10000$). Предполагается, что все города находятся на плоскости.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите одно вещественное число – наименьший радиус радиостанций. Число следует вывести с двумя знаками после запятой, без лидирующих нулей, в формате, приведенном в примерах.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 4<br>0 0<br>2 0<br>0 2<br>2 3 | 2.24 |
| 2 | 3<br>2 0<br>0 2<br>4 2 | 2.83 |

## 303. Цифры

*(Время: 1 сек. Память: 16 Мб Сложность: 27%)*

Составить программу, удаляющую одну цифру из N-значного числа, такую, чтобы плюс-минус сумма была наибольшей. Плюс-минус сумма – это сумма с чередованием цифр числа с разными знаками: для числа 764 это +7-6+4. Если удалить цифру 7, то будет +6-4=2, если удалить цифру 6, то будет +7-4=3, если удалить цифру 4, то будет +7-6=1. При этом видно, что максимум достигается при удалении средней цифры 6 и равен 3.

**Входные данные**

Во входном файле INPUT.TXT записано натуральное N-значное число ($2 \le N \le 50$).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите значение наибольшей суммы.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 764 | 3 |
| 2 | 6274861 | 14 |

## 304. Волейбол - 2

*(Время: 1 сек. Память: 16 Мб Сложность: 68%)*

Партия в волейболе, выигрывается командой, которая первой набирает 25 очков с преимуществом минимум в два очка. В случае равного счета 24-24, игра продолжается до достижения преимущества в 2 очка (26-24; 27-25). Исключение составляет лишь пятая партия (когда счет по партиям 2:2), в этом случае счет ведется по такому же принципу, но до 15 очков, а в случае счета 14:14 игра продолжается так же до достижения преимущества в 2 очка.

Две сыгранные партии, закончившиеся с одинаковым счетом, будем считать разными, если строки, в которые вписан порядок набора очков командами, не равны.

Комитет по проведению соревнований по волейболу заинтересовало, сколько различных партий может быть, заканчивающихся со счетом 25:23, оказывается 16123801841550, далее им стало интересно, сколько же существует различных матчей в которых первая команда победила в 3 партиях со счетом 25:23 25:20 25:18, оказывается 100431057869271076861662719709989250000.

Определить, сколько существует различных матчей, заканчивающихся заданным счетом. Два матча закончившиеся одинаковым количеством партий с одинаковым счетом, считаются различными, если есть различно сыгранные партии.

**Входные данные**

Во входном файле INPUT.TXT сначала записано число N – количество партий в матче. Далее следует N пар чисел, описывающих счет в каждой партии. При этом результаты партий разделяются пробелом, а счет в каждой партии отделяется двоеточием. Гарантируется, что счет в каждой партии соответствует возможному, согласно правилам волейбола, и ни в какой партии ни одна из команд не набирает более 40 очков.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите количество различных матчей, которые могут оканчиваться данным счетом.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3 25:23<br>25:20<br>25:18 | 100431057869271076861662719709989250000 |
| 2 | 4 25:23<br>20:25<br>26:24<br>25:18 | 32386609516427352165164579093098123021614066750000 |

## 305. Морской бой

*(Время: 1 сек. Память: 16 Мб Сложность: 60%)*

Участник игры в морской бой размещает на игровом поле свои корабли. По правилам этой разновидности игры корабли могут быть только прямоугольниками любых размеров, не могут пересекаться и иметь общих граничных точек. Количество уже размещённых кораблей равно K. Последний корабль он хочет сделать максимально большим.

**Входные данные**

В первой строке входного файла INPUT.TXT записаны три числа N, M и K – количество клеток по вертикали, количество клеток по горизонтали и число уже выстроенных кораблей соответственно ($1 \le N, M \le 100$, $1 \le K \le 10$). Следующие K строк содержат координаты K размещённых кораблей – 4 числа в каждой строке. 1-е и 2-е число – вертикальная и горизон-

тальная координаты левой верхней угловой клетки корабля, 3-е и 4-е число – вертикальная и горизонтальная координаты правой нижней угловой клетки корабля. Клетки поля нумеруются сверху вниз (от 1 до N) и слева направо (от 1 до M).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите одно число – количество клеток в последнем корабле.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 8 7 3<br>1 1 2 2<br>3 5 3 7<br>4 2 4 3 | 21 |

## 306. Танец

*(Время: 1 сек. Память: 16 Мб Сложность: 45%)*

На городском празднике танцуют девушки в красных и синих юбках. Они двигаются цепочкой и выполняют сложный рисунок танца. Из цепочки девушки выделяются по одной. Первая становится на левом краю сцены, вторая уходит в конец исходной цепочки, третья – на левый край сцены (справа от первой), четвертая – в конец исходной цепочки и т.д., пока все девушки не выстроятся на краю сцены.

Помогите постановщику танца определить, каким должно быть исходное расположение девушек, если на краю сцены, они выстроены так, что их юбки чередуются по цвету (слева направо): синяя, красная, синяя, красная и т.д.

**Входные данные**

Во входном файле INPUT.TXT записано натуральное число N – количество танцующих девушек ($N \le 1000$).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите строку, содержащую цепочку из N символов, состоящую из заглавных букв B и R, соответствующих цветам юбок – синему и красному.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 2 | BR |
| 2 | 3 | BBR |
| 3 | 4 | BBRR |

## 307. Атлеты

*(Время: 3 сек. Память: 16 Мб Сложность: 50%)*

Художественная гимнастика – это вид спорта, где всё познаётся в сравнении, здесь нельзя, как в беге или плавании, измерить результат спортсмена с точностью до сотой доли секунды. Поэтому на выступлениях оценки выступлениям дают судьи. При выставлении оценок судьи ориентируются не только на текущее выступление, но, безусловно, сравнивают текущее выступление с выступлениями, показанными ранее. Кроме того, учитывается сложность показанного упражнения и рейтинг спортсмена.

Каждый выход спортсмена описывается двумя числами: номер спортсмена в рейтинге (наиболее профессиональные спортсмены имеют наибольший номер), и номер исполненного упражнения (упражнения нумеруются, начиная с самых простых). Судья сравнивает каждый выход спортсмена с каждым из выполненных ранее выходов. Если в результате сравнения получается, что спортсмен с большим номером показал более простое упражнение, чем спортсмен с меньшим номером, судья удивляется. Следует учитывать, что один выход может удивить судью несколько раз. Один спортсмен может выполнить несколько выходов, так же,

как и одно упражнение может быть показано несколькими спортсменами – но такие выходы в сравнении судью не удивляют. Спортсмен не исполняет уже показанное упражнение повторно. Требуется подсчитать, сколько раз за время выступлений будет удивлён судья.

**Входные данные**

Первая строка входного файла INPUT.TXT содержит количество спортсменов N ($0 < N \le 250$), количество упражнений M ($0 < M \le 250$) и количество выходов P. Следующие P строк содержат по два числа, описывающие выход спортсмена – номер спортсмена и номер упражнения.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите, сколько раз был удивлен судья.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3 3 4<br>3 1<br>1 2<br>1 3<br>3 2 | 3 |
| 2 | 2 2 2<br>1 1<br>2 2 | 0 |

## 308. Вода

*(Время: 1 сек. Память: 16 Мб Сложность: 70%)*

Имеется три ведра, емкости которых известны и не равны. Самое большое ведро полное, остальные пусты. Требуется добиться, чтобы в самом большом ведре был заданный объем воды. За один шаг вода переливается из одного ведра в другое до тех пор, пока либо не закончится вода в ведре-источнике, либо не наполнится доверху вода в ведре-получателе.

Школьник Василий, чтобы занять себя, пытается решать эту задачу с разными входными данными, но не всегда находит решение. И даже если решение найдено, он хочет знать, является ли найденное решение оптимальным, а именно, используется ли минимальное количество шагов. Требуется написать программу, которая поможет Василию проверить его решение.

**Входные данные**

Во входном файле INPUT.TXT записаны 4 числа: емкости ведер B1, B2, B3 ($1000 \ge B1 > B2 > B3 > 0$) и требуемое количество воды T в первом ведре ($B1 > T > 0$).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите либо минимальное количество переливаний, либо если задача не имеет решения, то слово IMPOSSIBLE.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 10 8 4 4 | 3 |
| 2 | 10 8 4 5 | IMPOSSIBLE |

## 309. K-удивительные числа

*(Время: 3 сек. Память: 16 Мб Сложность: 30%)*

Переворотом числа X назовем число, в котором все цифры числа X стоят в обратном порядке. Например, переворотом числа 6372 является число 2736, а числа 7800 - 87. Назовем K-удивительным такое число, которое в сумме со своим переворотом дает число K.

Например, у числа 222 имеется всего два K-удивительных числа: 111 и 210, а у числа 1050 имеется девять K-удивительных числа: 129, 228, 527, 426, 525, 624, 723, 822, 921.

Требуется написать программу, которая по заданному K определит количество K-удивительных чисел.

**Входные данные**

Входной файл INPUT.TXT содержит одно натуральное число K ($1 \le K \le 10^6$).

**Выходные данные**

Выходной файл OUTPUT.TXT должен содержать одно число – количество K-удивительных чисел.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 222 | 2 |
| 2 | 1050 | 9 |

## 310. Рамка из клеток

*(Время: 1 сек. Память: 16 Мб Сложность: 33%)*

Прямоугольник состоит из X×Y квадратных клеток одинакового размера. Из него вырезан прямоугольник размером (X-2)×(Y-2) так, что осталась рамка шириной в одну клетку. Определить, можно ли покрыть всю рамку плитками размером A×1. Запас плиток неограничен, плитки не накладываются одна на другую и за пределы рамки не выходят.

Требуется написать программу, которая решает эту задачу.

**Входные данные**

Входной текстовый файл INPUT.TXT содержит в первой строке натуральное число K – количество тестов ($1 \le K \le 10$). В следующих K строках записаны по три натуральных числа: X, Y – размеры рамки, A – размер плитки ($3 \le X, Y \le 2\times10^9$, $1 \le A \le 2\times10^9$). Числа разделены пробелами.

**Выходные данные**

Выходной текстовый файл OUTPUT.TXT должен содержать одну строку из K символов 0 или 1 (1 – если покрытие существует, 0 – иначе).

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 1<br>3 3 1 | 1 |
| 2 | 2<br>3 3 2<br>3 3 3 | 10 |

## 311. Сумма факториалов

*(Время: 1 сек. Память: 16 Мб Сложность: 45%)*

Факториалом натурального числа K называется произведение K!=1×2×3×…×K.

Требуется написать программу, которая по заданному числу N вычислит сумму 1!+2!+…+N! .

**Входные данные**

Входной файл INPUT.TXT содержит одно натуральное число N ($N \le 200$).

**Выходные данные**

Выходной файл OUTPUT.TXT должен содержать все десятичные знаки искомой суммы.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 2 | 3 |
| 3 | 3 | 9 |

## 312. Арифметическая прогрессия

*(Время: 1 сек. Память: 16 Мб Сложность: 15%)*

Заданы первый и второй элементы арифметической прогрессии. Требуется написать программу, которая вычислит элемент прогрессии по ее номеру.

**Входные данные**

Входной файл INPUT.TXT содержит три целых числа, разделенных пробелами – первый элемент прогрессии $A_1$ ($1 \le A_1 \le 1000$), второй элемент прогрессии $A_2$ ($1 \le A_2 \le 1000$) и номер требуемого элемента N ($1 \le N \le 1000$).

**Выходные данные**

Выходной файл OUTPUT.TXT должен содержать одно целое число - N-й элемент арифметической прогрессии.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 1 5 3 | 9 |

## 313. Ежеминутные автобусы

*(Время: 1 сек. Память: 16 Мб Сложность: 30%)*

На автобусную остановку каждую минуту подходит автобус одного из маршрутов. Диспетчерская служба собрала данные за N минут – номера маршрутов каждого автобуса.

Требуется определить максимально возможное время ожидания для пассажира, желающего уехать определенным маршрутом. Т.е. в данной последовательности номеров маршрутов нужно найти два самых удаленных числа, равных между собой. Например, для последовательности 2, 11, 2, 2, 25, 11, 25, 11 максимальное время ожидания равно 4 (для маршрута номер 11).

**Входные данные**

Входной файл INPUT.TXT содержит в первой строке число N ($1 \le N \le 10^6$). Во второй строке записаны N чисел – номера маршрутов. Все числа натуральные и не превышают 100. Каждый номер маршрута встречается не менее двух раз.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите ответ на задачу.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 8<br>2 11 2 2 25 11 25 11 | 4 |
| 2 | 4<br>23 23 41 41 | 1 |

## 314. Лексикографический порядок чисел

*(Время: 1 сек. Память: 16 Мб Сложность: 31%)*

Натуральные числа от 1 до N упорядочены лексикографически. Например, для N=25 результат этого упорядочения будет таким: 1, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 2, 20, 21, 22, 23, 24, 25, 3, 4, 5, 6, 7, 8, 9.

Требуется написать программу, которая определит, на каком месте оказалось число K.

**Входные данные**

Входной файл INPUT.TXT содержит два натуральных числа N и K, записанных через пробел ($1 \le K \le N \le 10^4$).

**Выходные данные**

Выходной файл OUTPUT.TXT должен содержать одно натуральное число – номер места, на котором оказалось число K.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 25 17 | 9 |

## 315. Наименьшая система счисления

*(Время: 1 сек. Память: 16 Мб Сложность: 26%)*

Известно, что основанием позиционной системы счисления называют количество различных символов, используемых для записи чисел в данной системе счисления. Также известно, что любое число x в b-ичной системе счисления имеет вид $x=a_0 \cdot b^0+a_1 \cdot b^1+\ldots+a_n \cdot b^n$, где $b \geq 2$ и $0 \leq a_i < b$.

Для записи чисел в b-ичной системе счисления, где $b \leq 36$, могут быть использованы первые b символов из следующего списка 0, 1, …, 9, A, B, …, Z. Например, для записи чисел в троичной системы используются символы 0, 1, 2, а в двенадцатеричной - 0, 1, …, 9, A, B.

Требуется написать программу, которая по входной строке S определит, является ли данная строка записью числа в системе счисления, с основанием не большим 36, и, если является, определит минимальное основание этой системы счисления.

**Входные данные**

Входной файл INPUT.TXT содержит в единственной строке входную строку. Длина строки не превышает 255. Все символы строки имеют коды от 32 до 127.

**Выходные данные**

Выходной файл OUTPUT.TXT должен содержать одно число. Если строка является записью числа в некоторой системе счисления, то нужно вывести минимальное основание такой системы счисления. Иначе вывести -1.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 123 | 4 |
| 2 | ABCDEF | 16 |
| 3 | AD%AF | -1 |

## 316. Телеграфный перевод

*(Время: 1 сек. Память: 16 Мб Сложность: 29%)*

Телеграфный перевод оплачивается по 7 рублей за каждую полную и неполную сотню рублей. Например, за перевод 123 рублей надо заплатить 14 рублей – 7 рублей за полную сотню и 7 рублей за 23 рубля – неполную сотню. Некто попросил переслать ему зарплату в N рублей, взяв деньги за перевод из этой зарплаты.

Требуется написать программу, которая найдет, какую максимальную сумму некто сможет получить, и сколько денег будет стоить перевод.

**Входные данные**

Входной файл INPUT.TXT содержит одно натуральное число N ($8 \leq N \leq 6 \cdot 10^4$).

**Выходные данные**

Выходной файл OUTPUT.TXT должен содержать два натуральных числа – максимальную сумму и стоимость перевода. Числа разделить одним пробелом.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 10 | 3 7 |
| 2 | 108 | 100 7 |

## 317. Подарки Деда Мороза

*(Время: 1 сек. Память: 16 Мб Сложность: 27%)*

Ириска весит X грамм, мандарин – Y грамм, пряник – Z грамм.

Требуется написать программу, которая определит, сколько различных вариантов подарков весом ровно W грамм может сделать Дед Мороз.

**Входные данные**

В единственной строке входного файла INPUT.TXT содержится четыре числа X, Y, Z и W ($1 \leq X, Y, Z \leq 100$, $1 \leq W \leq 1000$).

**Выходные данные**

Выходной файл OUTPUT.TXT должен содержать одно целое число – количество вариантов подарков.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 10 25 15 40 | 3 |

## 318. Следующее число

*(Время: 1 сек. Память: 16 Мб Сложность: 36%)*

Задано натуральное число N.

Требуется написать программу, которая найдет следующее за ним число, в двоичном разложении которого столько же единиц, сколько в двоичном разложении числа N.

**Входные данные**

Входной файл INPUT.TXT содержит одно натуральное число N ($N \le 2^{30}$).

**Выходные данные**

Выходной файл OUTPUT.TXT должен содержать ответ на задачу.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 1 | 2 |
| 2 | 2 | 4 |
| 3 | 3 | 5 |

## 319. Точки отрезка

*(Время: 1 сек. Память: 16 Мб Сложность: 42%)*

Концы отрезка на плоскости имеют целочисленные координаты.

Требуется написать программу, которая вычислит, сколько всего точек с целочисленными координатами принадлежат этому отрезку.

**Входные данные**

Входной файл INPUT.TXT содержит четыре числа – координаты концов отрезка (x1, y1) и (x2, y2). Каждая из координат не превышает по абсолютной величине значения $10^9$.

**Выходные данные**

Выходной файл OUTPUT.TXT должен содержать одно число – количество точек на заданном отрезке, имеющих целочисленные координаты.

**Примеры**

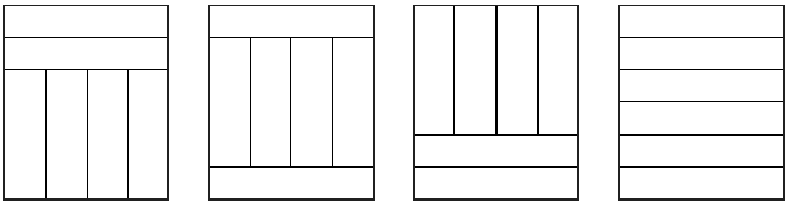
| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 1 1 2 2 | 2 |
| 2 | 0 0 -2 -2 | 3 |
| 3 | 1 1 1 10 | 10 |

## 320. Коридор

*(Время: 1 сек. Память: 16 Мб Сложность: 38%)*

Прямоугольный коридор длиной N метров и шириной M метров решили застелить N прямоугольными плитками шириной 1 метр и длиной M метров, таким образом, чтобы не было не застеленной поверхности.

Требуется написать программу, которая найдет количество способов это сделать. Например, для коридора с размерами 6 на 4 существует четыре способа застелить плитками 1 на 4.



**Входные данные**

Входной файл INPUT.TXT содержит два целых числа – M (длина плитки и ширина коридора) и N (длина коридора). Для этих чисел верны неравенства $2 \le M \le N \le 50$.

**Выходные данные**

Выходной файл OUTPUT.TXT должен содержать одно число – количество способов.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 4 6 | 4 |
| 2 | 2 2 | 2 |

## 321. Разные цифры

*(Время: 1 сек. Память: 16 Мб Сложность: 32%)*

Требуется написать программу, определяющую, в каких системах счисления с основаниями от 2 до 36 это число не содержит одинаковых цифр.

**Входные данные**

Входной файл INPUT.TXT содержит одно целое число N ($1 \le N \le 10^9$), записанное в десятичной системе счисления.

**Выходные данные**

Выходной файл OUTPUT.TXT должен содержать основания систем счисления в порядке возрастания, разделенные одним пробелом.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 100 | 11 12 13 14 15 16 17 18 20 21 22 23 25 26 27 28 29 30 31 32 33 34 35 36 |

## 322. Слово

*(Время: 1 сек. Память: 16 Мб Сложность: 28%)*

Числа Фибоначчи строятся следующим образом: первые два равны единице, а каждое следующее равно сумме двух предыдущих. Например, первые десять чисел Фибоначчи равны: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55. В заданном тексте символы нумеруются слева направо, начиная с единицы.

Требуется написать программу, которая составит слово из символов, номера которых совпадают с числами Фибоначчи.

**Входные данные**

Входной файл INPUT.TXT содержит в единственной строке текст, состоящий из латинских строчных букв. В тексте не более 30000 символов.

**Выходные данные**

Выходной файл OUTPUT.TXT должен содержать слово из символов, номера которых совпадают с числами Фибоначчи. Символы слова идут в том же порядке, что и в заданном тексте.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | a | a |
| 2 | abc | abc |
| 3 | abcdefghij | abceh |

## 323. Гипотеза Гольбаха

*(Время: 1 сек. Память: 16 Мб Сложность: 30%)*

Известно, что любое чётное число, большее 2, представимо в виде суммы 2 простых чисел, причём таких разложений может быть несколько. Впервые гипотезу о существовании данного разложения сформулировал математик Х. Гольбах.

Требуется написать программу, производящую согласно утверждению Гольбаха, разложение заданного чётного числа. Из всех пар простых чисел, сумма которых равна заданному числу, требуется найти пару, содержащую наименьшее простое число.

**Входные данные**

Входной файл INPUT.TXT содержит чётное число N ($4 \le N \le 998$).

**Выходные данные**

В выходной файл OUTPUT.TXT необходимо вывести два простых числа, сумма которых равна числу N. Первым выводится наименьшее число.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 6 | 3 3 |
| 2 | 992 | 73 919 |

**Разбор**

Для поиска такого разложения числа n достаточно перебрать всевозможные комбинации таких пар натуральных чисел a и b таких, что a+b=n и $a \le b$, и проверить их на простоту. Момент проверки числа на простоту в виде реализации функции IsPrime уже рассматривался здесь. Поэтому несложно оформить решение данной задачи в виде следующего алгоритма:

```
bool IsPrime(int n){
  int i;
  for i=2..sqrt(n)
   if(n mod i = 0) return false;
  return true;
}

read(n);

for i=2 .. n div 2
  if(IsPrime(i) and IsPrime(n-i)){
    write(i,' ',n-i);
    break;
  }
```

## 324. Четырехзначный палиндром

*(Время: 1 сек. Память: 16 Мб Сложность: 10%)*

Требуется написать программу, определяющую, является ли четырехзначное натуральное число N палиндромом, т.е. числом, которое одинаково читается слева направо и справа налево.

**Входные данные**

Входной файл INPUT.TXT содержит натуральное число N ($1000 \le N \le 9999$).

**Выходные данные**

В выходной файл OUTPUT.TXT следует вывести слово «YES», если число N является палиндромом, или «NO» – если нет.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 6116 | YES |
| 2 | 1231 | NO |

## 325. Мы с конем вдвоем по полю пойдем

*(Время: 1 сек. Память: 16 Мб Сложность: 31%)*

Заданы две клетки шахматной доски. Требуется определить, возможно ли попасть из одной клетки в другую одним ходом шахматного коня, а если нет, то следует выяснить, возможно ли попасть с помощью двух ходов.

**Входные данные**

Входной файл INPUT.TXT содержит координаты двух клеток в общепринятом формате: каждая координата записывается как латинская прописная буква и цифра, координаты отделены друг от друга запятой и пробелом.

**Выходные данные**

Выходной файл OUTPUT.TXT должен содержать цифру «1», если возможно из одной клетки в другую попасть за 1 ход, либо цифру «2», если попасть можно за 2 хода, либо «NO», если одна клетка недостижима из другой ни за 1 ни за 2 хода.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | a1, h8 | NO |
| 2 | a1, b3 | 1 |
| 3 | a1, d4 | 2 |

## 326. Преобразование последовательности - 2

*(Время: 1 сек. Память: 16 Мб Сложность: 29%)*

Задана последовательность, содержащая n целых чисел. Необходимо найти число, которое встречается в этой последовательности наибольшее количество раз, а если таких чисел несколько, то найти минимальное из них, и после этого переместить все такие числа в конец заданной последовательности. Порядок расположения остальных чисел должен остаться без изменения.

Например, последовательность 1, 2, 3, 2, 3, 1, 2 после преобразования должна превратиться в последовательность 1, 3, 3, 1, 2, 2, 2.

Требуется написать программу, которая решает данную задачу.

**Входные данные**

Первая строка входного файла INPUT.TXT содержит число n – количество чисел во входной последовательности ($3 \le n \le 100$). Следующая строка содержит входную последовательность, состоящую из n целых чисел, не превышающих по модулю 100. Все числа в строке разделены пробелом.

**Выходные данные**

В выходной файл OUTPUT.TXT выводится последовательность чисел, которая получается в результате названного преобразования. Все числа в последовательности должны быть разделены пробелом.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 7<br>1 2 3 2 3 1 2 | 1 3 3 1 2 2 2 |

## 327. В одном шаге от счастья

*(Время: 1 сек. Память: 16 Мб Сложность: 16%)*

Вова купил билет в трамвае 13-го маршрута и сразу посчитал суммы первых трёх цифр и последних трёх цифр номера билета (номер у билета шестизначный). Оказалось, что суммы отличаются ровно на единицу. «Я в одном шаге от счастья», – подумал Вова, – «или предыдущий или следующий билет точно счастливый». Прав ли он?

**Входные данные**

Входной файл INPUT.TXT содержит в первой строке число K – количество тестов. В следующих K строках записаны номера билетов. Количество тестов не больше 10. Номер состоит ровно из шести цифр, среди которых могут быть и нули. Гарантируется, что Вова умеет считать, то есть суммы первых трех цифр и последних трех цифр отличаются ровно на единицу.

**Выходные данные**

Выходной файл OUTPUT.TXT должен содержать K строк, в каждой из которых для соответствующего теста следует указать «Yes», если Вова прав, и «No», если нет.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3<br>715068<br>445219<br>012200 | Yes<br>No<br>Yes |

*Примечание*

Трамвайный билет называется счастливым, если сумма его первых трёх цифр равна сумме его последних трёх цифр.

## 328. Точки на костях

*(Время: 1 сек. Память: 16 Мб Сложность: 25%)*

Для того, чтобы заработать огромный капитал, новым русским необходимо иметь неординарное мышление. Конечно, при такой сложной работе, должны так же присутствовать какие то особенные механизмы для отдыха и развлечений. В этих целях в казино был придуман специальный набор домино для новых русских. Обычные кости домино представляют собой набор из различных комбинаций сочетаний двух плиток, на каждой из которых отображается от 0 до 6 точек. А этот набор представляет собой подобные сочетания плиток, но количество точек на каждой может быть от нуля до заданного значения, которое зависит от интеллектуального уровня игроков. В таком наборе костей присутствуют всевозможные сочетания плиток, но при этом ни одна из костей не повторяется (даже такие комбинации как 2-5 и 5-2 считаются одинаковыми).

Для изготовления данного набора костей перед изготовителем встала проблема вычисления суммарного количества точек на всех костях домино. Это связано с тем, что домино для новых русских украшается бриллиантами, которые представляют собой точки на плитках и при изготовлении необходимо оценить стоимость.

Помогите написать программу, которая решит эту задачу.

**Входные данные**

Входной файл INPUT.TXT содержит одно натуральное число N – максимальное количество точек на одной плитке домино ($N \leq 10000$).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите количество бриллиантовых камней, которые необходимо изготовить для заданного набора костей.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 2 | 12 |

## 329. Лесенка-2

*(Время: 1 сек. Память: 16 Мб Сложность: 37%)*

Вова стоит перед лесенкой из N ступеней. На каждой из ступеней написаны произвольные целые числа. Первым шагом Вова может перейти на первую ступень или, перепрыгнув через первую, сразу оказаться на второй. Также он поступает и дальше, пока не достигнет N-ой ступени. Посчитаем сумму всех чисел, написанных на ступенях через которые прошел Вова.

Требуется написать программу, которая определит оптимальный маршрут Вовы, при котором, шагая, он получит наибольшую сумму.

**Входные данные**

Входной файл INPUT.TXT содержит в первой строке натуральное число N – количество ступеней лестницы. Во второй строке через пробел заданы числа, написанные на ступенях лестницы, начиная с первой. Количество ступеней не превышает 1000, числа, написанные на ступенях, не превосходят по модулю 1000.

**Выходные данные**

Выходной файл OUTPUT.TXT должен содержать в первой строке наибольшее значение суммы. Во второй строке должны быть записаны через пробел номера ступеней по возрастанию, по которым должен шагать Вова.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3<br>1 2 1 | 4<br>1 2 3 |
| 2 | 3<br>1 -1 1 | 2<br>1 3 |

## 330. Телепортация

*(Время: 1 сек. Память: 16 Мб Сложность: 30%)*

Вова попал на космическом корабле на бесконечную плоскую планету в точку с координатами ($x_1$, $y_1$). Вова, управляя кораблем, имеет возможность за одну секунду телепортироваться из точки (x, y) в одну из точек (x+C, y+C), (x+C, y-C), (x-C, y+C), (x-C, y-C), где C - произвольное натуральное число.

Требуется написать программу, которая определит, через какое минимальное время Вове удастся достичь точки ($x_2$, $y_2$).

**Входные данные**

Входной файл INPUT.TXT содержит в первой строке числа $x_1$, $y_1$, во второй – $x_2$, $y_2$. Все числа целые от нуля до $10^6$. Точки ($x_1$, $y_1$) и ($x_2$, $y_2$) не совпадают.

**Выходные данные**

Выходной файл OUTPUT.TXT должен содержать одно число – минимальное время телепортации. Если такая телепортация невозможна, то вывести 0.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 0 0<br>0 2 | 2 |

## 331. Время прибытия

*(Время: 1 сек. Память: 16 Мб Сложность: 26%)*

Задано время отправления поезда и время в пути до конечной станции. Требуется написать программу, которая найдет время прибытия этого поезда (возможно, в другие сутки).

**Входные данные**

Входной файл INPUT.TXT содержит две строки. В первой строке задано время отправления, а во второй строке – время в пути. Время отправления задается в формате «HH:MM», где HH время в часах, которое принимает значение от 00 до 23, MM – время в минутах, которое принимает значение от 00 до 59. Время в пути задается двумя неотрицательными целыми числами – количество часов и количество минут. Числа разделяются одним пробелом. Количество часов не превышает 120, минут – 59.

**Выходные данные**

Выходной файл OUTPUT.TXT должен содержать одну строку – время прибытия поезда на конечную станцию. Формат вывода этого времени совпадает с форматом ввода времени отправления.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 00:00<br>10 10 | 10:10 |
| 2 | 01:02<br>4 6 | 05:08 |
| 3 | 11:00<br>22 0 | 09:00 |

## 332. Минимальная стоимость проезда

*(Время: 1 сек. Память: 16 Мб Сложность: 40%)*

На прямой ветке железной дороги расположено несколько станций. Задана стоимость проезда между любыми двумя станциями.

Требуется написать программу нахождения минимальной стоимости проезда между крайними станциями. Двигаться по железной дороге можно только в одном направлении (от станции с меньшим номером до станции с большим номером.).

**Входные данные**

Входной файл INPUT.TXT содержит в первой строке натуральное число N, не большее 250. Всего на дороге расположено N+1 станций, пронумерованных от 0 до N. В следующих строках записано N(N+1)/2 чисел, задающих стоимости проезда между станциями: сначала стоимость проезда от станции 0 до станций 1, 2, 3, …, N, затем от станции 1 до станций 2, 3, …, N, …, от станции N-1 до станции N. Все стоимости проезда – натуральные числа, не превосходящие 10000.

**Выходные данные**

Выходной файл OUTPUT.TXT должен содержать одно число – минимальную стоимость проезда от станции 0 до станции N с возможными пересадками.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3<br>7 10 20<br>4 8<br>2 | 12 |

**Пояснение**

В приведенном примере всего 4 станции с номерами 0, 1, 2, 3. Оптимальный маршрут проходит через станции 0, 2 и 3. Его стоимость равна 10+2=12.

## 333. Общие цифры

*(Время: 1 сек. Память: 16 Мб Сложность: 28%)*

Заданы натуральные числа A, B, C. Требуется написать программу, которая найдет общие цифры в этих числах.

**Входные данные**

Входной файл INPUT.TXT содержит три натуральных числа A, B, C ($1 \le A, B, C \le 10^{80}$). Числа разделены одним пробелом.

**Выходные данные**

Выходной файл OUTPUT.TXT должен содержать в первой строке количество общих цифр, а во второй строке в порядке возрастания через один пробел общие цифры.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 1 2 3 | 0 |
| 2 | 12 13 14 | 1<br>1 |
| 3 | 1234 2345 3456 | 2<br>3 4 |

## 334. Китайские часы

*(Время: 1 сек. Память: 16 Мб Сложность: 37%)*

Русский бизнесмен Иван Петров закупил в Китае большую партию наручных часов, чтобы продать их на родине за полцены (т.е. в 5 раз дороже, чем они стоили в Китае). Иван столкнулся с проблемой: китайские часы оказались некачественными. Мало того, что часы работали на протяжении всего нескольких часов, пока их не стукнешь, так еще и время под-

водить неудобно: вращать можно не минутную, а только секундную стрелку, причем, что самое ужасное, только в одну сторону в направлении увеличении времени. Например, для того, чтобы подвести часы на секунду назад, необходимо было сделать более 700 полных оборотов секундной стрелки, на что Иван бы потратил более 10 минут.

Чтобы продать эти часы оптом Ивану необходимо на момент сделки создать видимость того, что часы исправны. Для этого он собирается остановить все часы, установить их на одно и то же время. А перед сделкой ударить по чемодану с часами, чтобы они все дружно пошли.

Помогите Ивану выяснить: какое время на часах лучше установить для того, чтобы Иван потратил как можно меньше времени для того, чтобы подвести все часы.

**Входные данные**

В первой строке входного файла INPUT.TXT содержится натуральное число N – количество часов ($N \le 50000$). В последующих N строках располагаются показания всех часов в формате h:mm:ss, где h – показывает который час, mm – минуты, ss - часы ($1 \le h \le 12$, $0 \le mm \le 59$, $0 \le ss \le 59$).

**Выходные данные**

Выходной файл OUTPUT.TXT должен содержать время, которое нужно установить на всех часах, в формате, указанном выше. В случае неоднозначного ответа выведите наименьшее время.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3<br>8:19:16<br>2:05:11<br>12:50:07 | 2:05:11 |

## 335. Трипростые числа

*(Время: 1 сек. Память: 16 Мб Сложность: 40%)*

Будем называть натуральное число *трипростым*, если в нем любые подряд идущие 3 цифры образуют трехзначное простое число.

Требуется найти количество N-значных трипростых чисел.

**Входные данные**

Входной файл INPUT.TXT содержит натуральное число N ($3 \le N \le 10000$).

**Выходные данные**

Выходной файл OUTPUT.TXT должен содержать количество N-значных трипростых чисел, которое следует вывести по модулю $10^9+9$.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 4 | 204 |

## 336. Лифт

*(Время: 1 сек. Память: 16 Мб Сложность: 20%)*

В доме Вилли установили скоростной лифт новой экспериментальной модели. В этом лифте кнопки с номерами этажей заменены двумя другими кнопками. При нажатии на первую кнопку лифт поднимается на один этаж вверх, а при нажатии на вторую – опускается на один этаж вниз.

Младшему брату Вилли Дилли очень нравится кататься на новом лифте. Он катается на нём до тех пор, пока не побывает на каждом из этажей хотя бы по одному разу. После этого Дилли довольный возвращается домой.

Зная порядок, в котором Дилли нажимал на кнопки лифта, попробуйте определить общее количество этажей в доме Вилли и Дилли.

**Входные данные**

Первая строка входного файла INPUT.TXT содержит последовательность нажатий на кнопки лифта. Символ «1» означает, что была нажата первая кнопка, а символ «2» – что была нажата вторая кнопка. Символы «1» и «2» не разделены пробелами. Количество нажатий не превосходит 100. Гарантируется, что лифт никогда не опускался ниже первого и не поднимался выше последнего этажа.

**Выходные данные**

В выходной файл OUTPUT.TXT следует вывести одно число – количество этажей в доме Вили и Дилли.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 11 | 3 |
| 2 | 21212 | 2 |
| 3 | 1221221221221 | 6 |

## 337. Лампочки

*(Время: 1 сек. Память: 16 Мб Сложность: 94%)*

Имеется ряд из N лампочек, которые пронумерованы от 1 до N. Изначально ни одна из лампочек не горит. Далее происходит K последовательных линейных инверсий этого ряда ламп. Под линейной инверсией понимается инверсия каждой P-ой лампочки в ряде. Например, если P=3, то произойдет инверсия 3-ей, 6-ой, 9-ой и т.д. лампочек.

Требуется определить, сколько горящих лампочек останется после реализации всех заданных линейных инверсий?

**Входные данные**

В первой строке входного файла INPUT.TXT заданны числа N и K – число лампочек и число линейных инверсий. Вторая строка состоит из K целых чисел Pi, задающих период данных инверсий ($1 \le N \le 10^9$, $1 \le K \le 100$, $1 \le P_i \le 50$).

**Выходные данные**

В выходной файл OUTPUT.TXT следует вывести ответ на задачу.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 20 3<br>2 3 8 | 8 |
| 2 | 172 10<br>19 2 7 13 40 23 16 1 45 9 | 99 |

## 338. Лоскутки

*(Время: 1 сек. Память: 16 Мб Сложность: 50%)*

Вася Пупкин взял листочек в клетку и начал его резать по определённым линиям. На запасном листке такого же размера он закрасил клетки, по которым проходили линии. Василий Васильевич так увлёкся этим занятием, что запутался, сколько частей от листа у него осталось. Ваша задача это число.

**Входные данные**

Во входном файле INPUT.TXT в первой строке записаны N и M ($0 < N, M \le 100$) – размерность матрицы. Далее записана матрица из N строк, каждая из которых содержит M нулей и единиц. 0 обозначает не закрашенную клетку и 1 – закрашенную (линию разреза).

**Выходные данные**

В выходной файл OUTPUT.TXT следует вывести количество оставшихся частей листа.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 4 4<br>0 0 1 0<br>0 0 1 0<br>0 1 0 0<br>1 0 0 1 | 2 |

## 339. Мероприятие

*(Время: 1 сек. Память: 16 Мб Сложность: 31%)*

Мише поручили решить следующую задачу: имеется дата начала и конца некоторого мероприятия; требуется определить его длительность. Он написал программу и попросил Машу её проверить.

Через некоторое время пришла Маша и расстроила Мишу: «Твоя программа работает неправильно. По-моему, ты забыл, что года бывают високосными».

У Миши очень мало свободного времени и он не успевает исправить свою программу. Помогите ему.

Год является високосным, тогда и только тогда, когда выполнено одно из следующих условий:

- год делится на 4, но не делится на 100;
- год делится на 400.

Например, года 400, 404, 496, 504, 2000, 2004 являются високосными, а года 100, 200, 300, 503, 1000, 2001, 2005 – нет.

**Входные данные**

Первая строка входного файла INPUT.TXT содержит дату начала мероприятия. Вторая строка входного файла содержит дату конца мероприятия. Гарантируется, что первая дата меньше второй. Даты заданы в формате DD.MM.YYYY.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите единственное число - длительность мероприятия (в днях).

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 01.09.2005<br>06.10.2005 | 36 |
| 2 | 01.09.0005<br>01.09.0006 | 366 |
| 3 | 01.02.2004<br>01.03.2004 | 30 |

## 340. Коробки

*(Время: 1 сек. Память: 16 Мб Сложность: 19%)*

На столе лежат коробка размера A1 × B1 × C1 и коробка размера A2 × B2 × C2. Выясните можно ли одну из этих коробок положить в другую, если разрешены повороты коробок вокруг любого ребра на угол 90 градусов.

**Входные данные**

Первая строка входного файла содержит три целых числа A1, B1 и C1. Вторая строка входного файла содержит три целых числа A2, B2 и C2. Все числа положительны и не превосходят 1000.

**Выходные данные**

Если коробки одинаковы, выведите «Boxes are equal». Если первая коробка может быть положена во вторую, выведите «The first box is smaller than the second one». Если вторая коробка может быть положена в первую, выведите «The first box is larger than the second one». Иначе, выведите «Boxes are incomparable».

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 1 2 3<br>3 2 1 | Boxes are equal |
| 2 | 2 2 3<br>3 2 1 | The first box is larger than the second one |
| 3 | 2 2 3<br>3 2 3 | The first box is smaller than the second one |
| 4 | 3 4 5<br>2 4 6 | Boxes are incomparable |

## 341. Числовая последовательность

*(Время: 1 сек. Память: 16 Мб Сложность: 35%)*

Дима недавно поступил на работу в научно-исследовательский институт «Числовые Последовательности». Как следует из названия этого института, основным направлением его работы является проведение различных исследований в области числовых последовательностей.

Недавно руководитель отдела, где начал работать Дима, при решении одной из проблем столкнулся с весьма интересной последовательностью чисел $a_1$, $a_2$, …, $a_n$, …, которая определяется следующим образом: $a_1 = 0$ и каждое последующее число $a_i$ ($1 < i \le n$) определяется как наименьшее большее натуральное число, десятичная запись которого не содержит цифр, представленных в десятичной записи $a_{i-1}$.

Требуется написать программу, которая по значению числа n вычисляет величину $a_n$.

**Входные данные**

Входной файл INPUT.TXT содержит натуральное число N ($N \le 500$).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите искомое число $a_N$.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 1 | 0 |
| 2 | 28 | 911 |

## 342. Вписанная окружность

*(Время: 1 сек. Память: 16 Мб Сложность: 68%)*

Очень интересными объектами, которые изучаются в планиметрии, являются вписанные и описанные окружности. Известно, например, что вокруг любого треугольника можно описать окружность и в любой треугольник можно вписать окружность. А что будет, если вместо треугольника задан выпуклый многоугольник?

Требуется написать программу, которая определяет, можно ли в заданный выпуклый многоугольник вписать окружность, и, если это можно сделать, то вычисляет координаты ее центра и радиус.

**Входные данные**

Первая строка входного файла INPUT.TXT содержит количество вершин многоугольника n ($3 \le n \le 8$). Последующие n строк содержат координаты вершин многоугольника в порядке обхода против часовой стрелки, каждая i-ая из них содержит два целых числа: $x_i$ и $y_i$, значения которых не превосходят 1000 по абсолютной величине.

**Выходные данные**

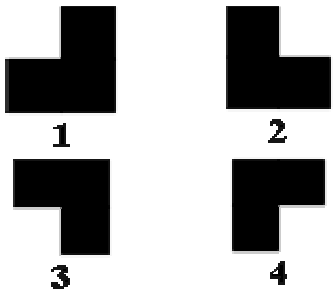
В первой строке выходного файла OUTPUT.TXT необходимо вывести YES, если окружность, вписанная в заданный многоугольник, существует, в противном случае следует вывести слово NO. В случае положительного ответа во второй строке следует указать координаты центра окружности и ее радиус. Числа следует выводить с тремя знаками после точки (даже если там нули), округляя их до $10^{-3}$ по математическим правилам. Число «0.000» следует выводить без знака «-».

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 4<br>0 0<br>1 0<br>1 1<br>0 1 | YES<br>0.500 0.500 0.500 |
| 2 | 4<br>0 0<br>1 0<br>1 2<br>0 2 | NO |

## 343. Укладка плиток

*(Время: 1 сек. Память: 16 Мб Сложность: 35%)*




Вы являетесь одним из разработчиков нового архитектурного пакета прикладных программ «CadArch». Одной из его функций является проектирование укладки половых плиток. В настоящее время вы занимаетесь программной реализацией модуля, который отвечает за укладку плиток в прямоугольных помещениях.

Для простоты будем считать, что пол помещения представляет собой прямоугольник размером n на m метров, разбитый на m·n квадратиков со стороной по 1 метру. Кроме этого, будем считать, что имеется четыре типа плиток, показанные в таблице. Каждая из плиток представляет собой квадрат размером 2 на 2 метра, из которого вырезан один квадратик размером 1 на 1 метр.

Проектируемый модуль должен работать следующим образом. На вход модуля подается набор команд, каждая из которых обозначает, в какое место и какого типа плитку необходимо положить. Команда обрабатывается следующим образом: если ни один из квадратиков, который должна занимать текущая плитка, не занят и плитка полностью помещается внутри прямоугольника, то плитка размещается в указанном месте, в противном случае – нет.

Требуется написать программу, которая определяет, какая площадь в соответствии с заданным набором команд будет покрыта плитками.

**Входные данные**

Первая строка входного файла INPUT.TXT содержит два числа n и m – высота и ширина пола помещения ($1 \le m, n \le 50$). Вторая строка содержит число k – количество команд, которые необходимо обработать. Каждая из последующих k строк содержит описание одной команды из набора команд. Описание команды состоит из трех чисел. Первое число определяет тип плитки (число от 1 до 4), а два других - координаты левого верхнего квадрата (y, x) размером 2 на 2, в который вписана соответствующая плитка ($0 \le x, y, k \le 1000$).

**Выходные данные**

В выходной файл OUTPUT.TXT необходимо вывести одно число, определяющее площадь, покрытую плитками после выполнения заданной во входном файле последовательности команд.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 4 4<br>4<br>4 1 1<br>2 2 2<br>3 1 1<br>1 3 3 | 9 |

## 344. Ближайшие точки

*(Время: 1 сек. Память: 16 Мб Сложность: 38%)*

Антон в школе начал изучать математику. Его внимание привлекло новое для него понятие числовой прямой. Антон быстро научился вычислять расстояния между двумя точками на этой прямой, задавать отрезки и интервалы на ней.

Готовясь к контрольной работе, Антон столкнулся со следующей задачей: «На числовой прямой задано n точек. Необходимо найти среди них две ближайшие». Расстояние между двумя точками числовой прямой x и y равно |x - y|.

Требуется написать программу, которая поможет Антону решить поставленную задачу.

**Входные данные**

Первая строка входного файла INPUT.TXT содержит количество точек n ($2 \le n \le 10^5$). Вторая строка содержит n различных целых чисел $x_i$ – координаты заданных точек числовой прямой. Числа в строке разделены пробелом. Значения всех координат $x_i$ не превосходят $10^9$ по абсолютной величине.

**Выходные данные**

В первой строке выходного файла OUTPUT.TXT необходимо вывести минимальное расстояние между двумя точками, заданными во входном файле. Во второй строке выходного файла необходимо вывести номера точек, которым соответствует найденное расстояние. Точки нумеруются натуральными числами от 1 до n в том порядке, в котором они заданы во входном файле. Если ответов несколько, выведите тот из них, в котором точки расположены левее других на числовой прямой. Первым выводится номер левой точки, далее через пробел – номер правой точки.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 5<br>10 3 6 2 5 | 1<br>4 2 |

## 345. Рекурсия

*(Время: 1 сек. Память: 16 Мб Сложность: 46%)*

Одним из важных понятий, используемых в теории алгоритмов, является рекурсия. Неформально ее можно определить как использование в описании объекта самого себя. Если речь идет о процедуре, то в процессе исполнении эта процедура напрямую или косвенно (через другие процедуры) вызывает сама себя.

Рекурсия является очень «мощным» методом построения алгоритмов, но таит в себе некоторые опасности. Например, неаккуратно написанная рекурсивная процедура может войти в бесконечную рекурсию, то есть, никогда не закончить свое выполнение (на самом деле, выполнение закончится с переполнением стека).

Поскольку рекурсия может быть косвенной (процедура вызывает сама себя через другие процедуры), то задача определения того факта, является ли данная процедура рекурсивной, достаточно сложна. Попробуем решить более простую задачу.

Рассмотрим программу, состоящую из n процедур $P_1$, $P_2$, …, $P_n$. Пусть для каждой процедуры известны процедуры, которые она может вызывать. Процедура P называется потенциально рекурсивной, если существует такая последовательность процедур $Q_0$, $Q_1$, …, $Q_k$, что $Q_0 = Q_k = P$ и для i = 1…k процедура $Q_{i-1}$ может вызвать процедуру $Q_i$. В этом случае задача будет заключаться в определении для каждой из заданных процедур, является ли она потенциально рекурсивной.

Требуется написать программу, которая позволит решить названную задачу.

**Входные данные**

Первая строка входного файла INPUT.TXT содержит целое число n – количество процедур в программе ($1 \le n \le 100$). Далее следуют n блоков, описывающих процедуры. После каждого блока следует строка, которая содержит 5 символов «*».

Описание процедуры начинается со строки, содержащий ее идентификатор, состоящий только из маленьких букв латинского алфавита и цифр. Идентификатор непуст, и его длина не превосходит 100 символов. Далее идет строка, содержащая число k ($k \le n$ ) – количество процедур, которые могут быть вызваны описываемой процедурой. Последующие k строк содержат идентификаторы этих процедур – по одному идентификатору на строке.

Различные процедуры имеют различные идентификаторы. При этом ни одна процедура не может вызвать процедуру, которая не описана во входном файле.

**Выходные данные**

В выходной файл OUTPUT.TXT для каждой процедуры, присутствующей во входных данных, необходимо вывести слово YES, если она является потенциально рекурсивной, и слово NO – в противном случае, в том же порядке, в каком они перечислены во входных данных.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3<br>p1<br>2<br>p1<br>p2<br>*****<br>p2<br>1<br>p1<br>*****<br>p3<br>1<br>p1<br>***** | YES<br>YES<br>NO |

## 346. Сумма двух чисел

*(Время: 1 сек. Память: 16 Мб Сложность: 46%)*

Заданы три числа: a, b, c. Необходимо выяснить, можно ли так переставить цифры в числах a и b, чтобы в сумме получилось c.

**Входные данные**

Входной файл INPUT.TXT содержит три целых числа: a, b, c ($0 < a, b, c < 10^9$). Числа разделены пробелом.

**Выходные данные**

В выходной файл OUTPUT.TXT следует вывести YES, если искомая перестановка цифр возможна, в противном случае необходимо вывести NO. При положительном ответе во второй строке следует вывести число x, получаемое перестановкой цифр числа a, и число y, получаемое перестановкой цифр числа b, сумма которых равна c. Числа x и y не должны содержать ведущих нулей. Числа в строке разделены пробелом. Если решений несколько, то следует вывести ту пару, в которой число x минимально.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 12 31 25 | YES<br>12 13 |
| 2 | 12 31 26 | NO |

## 347. Покер

*(Время: 1 сек. Память: 16 Мб Сложность: 33%)*

Имеется 5 целых чисел. Среди них:

- если одинаковы 5, то вывести «Impossible», иначе
- если одинаковы 4, то вывести «Four of a Kind», иначе

• если одинаковы 3 и 2, то вывести «Full House», иначе
• если есть 5 последовательных, то вывести «Straight», иначе
• если одинаковы 3, то вывести «Three of a Kind», иначе
• если одинаковы 2 и 2, то вывести «Two Pairs», иначе
• если одинаковы 2, то вывести «One Pair», иначе
• вывести «Nothing».

**Входные данные**

Входной файл INPUT.TXT содержит 5 целых чисел от 1 до 13, разделенных пробелом.

**Выходные данные**

В выходной файл OUTPUT.TXT следует вывести результат анализа.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 1 3 9 3 2 | One Pair |
| 2 | 1 5 5 4 4 | Two Pairs |
| 3 | 1 5 2 4 3 | Straight |
| 4 | 10 11 12 13 1 | Nothing |

## 348. Пересечение отрезков

*(Время: 1 сек. Память: 16 Мб Сложность: 49%)*

Два отрезка на плоскости заданы целочисленными координатами своих концов в декартовой системе координат. Требуется определить, существует ли у них общая точка.

**Входные данные**

Входной файл INPUT.TXT содержит координаты четырех точек, задающих отрезки. В первой строке содержатся координаты первого конца первого отрезка, во второй – второго конца первого отрезка, в третьей и четвёртой – координаты концов второго отрезка. Все координаты – целые числа, не превосходящие 10000 по абсолютной величине.

**Выходные данные**

В выходной файл OUTPUT.TXT следует вывести слово «Yes», если общая точка есть, или слово «No» – в противном случае.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 0 0<br>1 1<br>0 1<br>1 0 | Yes |
| 2 | 0 0<br>1 0<br>0 1<br>1 1 | No |

## 349. Простые числа

*(Время: 1 сек. Память: 16 Мб Сложность: 28%)*

Необходимо вывести все простые числа от M до N включительно.

**Входные данные**

Входной файл INPUT.TXT содержит два натуральных числа M и N, разделенных пробелом ($2 \le M \le N \le 10^6$).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите в одной строке через пробел все простые числа от M до N в порядке возрастания. Если таковых чисел нет, то следует вывести «Absent».

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 2 5 | 2 3 5 |
| 2 | 4 4 | Absent |

**Разбор**

Масса олимпиадных задач связаны с простыми числами, в частности с их поиском и определением простоты. Поэтому, прежде чем начинать разбор данной задачи, поговорим сначала о том, как определить простоту конкретного числа. Известно, что если число n составное, то один из его делителей не превосходит значения sqrt(n), поэтому при проверке на делимость числа n достаточно перебрать всевозможные делители от 2 до sqrt(n). Это можно оформить в виде следующей функции:

```
bool IsPrime(int n){
  int i;
  for i=2..sqrt(n)
   if(n mod i = 0) return false;
  return true;
}
```

Описанную выше функцию можно ускорить вдвое, если проверять делимость только на нечетные числа, а делимость на 2 рассмотреть отдельно. Так же когда уже известны все простые числа, не превосходящие sqrt(n), то еще более разумно проверять делимость только на них, тогда скорость возрастет в ln(sqrt(n)) раз. Следующий фрагмент программы демонстрирует заполнение массива p[i] простыми числами от 2 до n:

```
  p[1]=2; np=1;              //в массив p вносим первое простое число 2 (всего
np чисел)
  for x=3..n{                //цикл по нечетным значениям x, которые проверяем
на простоту
    j=1; Ok=true;
    while(p[j]*p[j]<=x)      //проверяем число x на делимость на возможные ран-
нее найденые простые числа
      if(x mod p[j] = 0){
        Ok=false; break;
      }
    if(Ok){                  //если делитель не найден, то добавляем данное чис-
ло в список простых
      np=np+1;
      p[np]=x;
    }
  }
```

Для того, чтобы довести вышеописанный алгоритм до алгоритма решения исходной задачи достаточно в процессе поиска проверять добавляемое простое число на попадение в заданный отрезок и выводить его в случае принадлежности. И не забудьте про двойку!

## 350. Перестановки

*(Время: 1 сек. Память: 16 Мб Сложность: 44%)*

Дана строка, состоящая из N попарно различных символов. Требуется вывести все перестановки символов данной строки в алфавитном порядке.

**Входные данные**

Входной файл INPUT.TXT содержит строку, состоящую не более чем из N символов ($1 \le N \le 8$).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите все элементы искомой последовательности по одному в каждой строке.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | AB | AB<br>BA |

| | | |
|---|---|---|
| 2 | 123 | 123<br>132<br>213<br>231<br>312<br>321 |

## 351. Прыжки по буквам

*(Время: 1 сек. Память: 16 Мб Сложность: 56%)*

Дана цепочка из N символов, состоящая из прописных латинских букв. Необходимо пройти с первого символа цепочки до последнего символа, прыгая не более чем на K символов. Стоимость прыжка, при котором символ не меняется, равна 0, а стоимость прыжка на другой символ равна 1.

Требуется написать программу, которая вычислит наименьшую стоимость перехода с первого на последний символ.

**Входные данные**

Входной файл INPUT.TXT содержит в первой строке два целых числа: длина цепочки N ($2 \le N \le 10^5$) и максимальная длина прыжка K ($1 \le K < N$). Во второй строке содержится цепочка из N прописных латинских букв.

**Выходные данные**

Выходной файл OUTPUT.TXT должен содержать одно число – минимальную стоимость перехода.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 10 2<br>ABABBCACBC | 2 |

## 352. Дробь

*(Время: 1 сек. Память: 16 Мб Сложность: 39%)*

Вася учится в третьем классе и сейчас он проходит тему «Простые дроби с натуральными числителем и знаменателем». Оказывается, что дробь называется правильной, если ее числитель меньше знаменателя, и несократимой, если числитель и знаменатель являются взаимно простыми. Вася очень любит математику и поэтому дома он решает много задач. В данный момент Вася ищет наибольшую правильную несократимую дробь, у которой сумма числителя и знаменателя равна N.

Требуется написать программу, которая поможет Васе решить эту задачу.

**Входные данные**

Входной файл INPUT.TXT содержит одно целое число N ($3 \le N \le 2 \cdot 10^9$).

**Выходные данные**

Выходной файл OUTPUT.TXT должен содержать два числа – числитель и знаменатель найденной дроби, разделенные пробелом.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3 | 1 2 |
| 2 | 10 | 3 7 |

## 353. Треугольники

*(Время: 1 сек. Память: 16 Мб Сложность: 41%)*

Дан набор из нескольких отрезков. Необходимо составить треугольник наибольшей площади, используя в качестве сторон три отрезка из заданных.

Требуется написать программу, которая найдет наибольшую площадь треугольника.

**Входные данные**

Входной файл INPUT.TXT содержит в первой строке одно целое число N ($3 \le N \le 1000$) – количество отрезков. Во второй строке содержатся N целых чисел от 1 до 1000 – длины отрезков. Числа разделены пробелом.

**Выходные данные**

Выходной файл OUTPUT.TXT должен содержать одно число с тремя десятичными знаками после запятой – наибольшую площадь треугольника из заданных отрезков. Если из заданных отрезков нельзя построить ни одного треугольника, то вывести в файл 0.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 5<br>2 4 8 16 7 | 13.998 |
| 2 | 3<br>3 4 5 | 6.000 |
| 3 | 3<br>1 2 5 | 0 |

## 354. Разложение на простые множители

*(Время: 1 сек. Память: 16 Мб Сложность: 27%)*

Требуется вывести представление целого числа N в виде произведения простых чисел.

**Входные данные**

Входной файл INPUT.TXT содержит натуральное число N ($2 \le N \le 2^{31}-1$).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите список простых множителей числа N в порядке неубывания, разделенных знаком «*».

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 5 | 5 |
| 2 | 30 | 2*3*5 |

## 355. Перестановки - 2

*(Время: 1 сек. Память: 16 Мб Сложность: 46%)*

Дана строка, состоящая из N символов. Требуется вывести все перестановки символов данной строки в лексикографическом порядке без повторений.

**Входные данные**

Входной файл INPUT.TXT содержит N - количество символов в строке ($1 \le N \le 8$).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите все элементы искомой последовательности по одному в каждой строке.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | AB | AB<br>BA |
| 2 | 122 | 122<br>212<br>221 |

## 356. Копилка

*(Время: 1 сек. Память: 16 Мб Сложность: 49%)*

Задан вес E пустой копилки и вес F копилки с монетами. В копилке могут находиться монеты N видов, для каждого вида известна ценность Pi и вес Wi одной монеты. Найти минимальную и максимальную суммы денег, которые могут находиться в копилке.

**Входные данные**

В первой строке входного файла INPUT.TXT находятся числа E и F, во второй - число N, в следующих N строках – по два числа, Pi и Wi. ($1 \le E \le F \le 10000$, $1 \le N \le 500$, $1 \le Pi \le 50000$, $1 \le Wi \le 10000$).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите два числа через пробел – минимальную и максимальную суммы. Если копилка не может иметь точно заданный вес при условии, что она наполнена монетами заданных видов, следует вывести «This is impossible.».

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 1000 1100<br>2<br>1 1<br>5 2 | 100 250 |
| 2 | 1000 1010<br>2<br>6 3<br>2 2 | 10 16 |
| 3 | 1000 2000<br>1<br>10 3 | This is impossible. |

## 357. Делимость на 11

*(Время: 1 сек. Память: 16 Мб Сложность: 22%)*

Для делимости числа на 11 необходимо, чтобы разность между суммой цифр, стоящих на четных местах, и суммой цифр, стоящих на нечетных местах, делилась на 11.

Требуется написать программу, которая проверит делимость заданного числа на 11.

**Входные данные**

Входной файл INPUT.TXT содержит одно натуральное число N, делимость которого надо проверить ($1 \le N \le 10^{10000}$).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите «YES», если число делится на 11, или «NO» иначе.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 121 | YES |
| 2 | 1211 | NO |

## 358. Забор в парке

*(Время: 1 сек. Память: 16 Мб Сложность: 42%)*

В бесконечном парке деревья образуют квадратную решетку с шагом один метр. Часть парка было решено оградить забором, который представляет собой треугольник с заданными координатами вершин. Деревья, которые в точности попадают на вершины или стороны треугольника, придется срубить.

Требуется написать программу, которая найдет количество таких деревьев.

**Входные данные**

Входной файл INPUT.TXT содержит шесть целых чисел – координаты вершин треугольника (абсцисса, ордината). Все числа по абсолютной величине не превышают $10^9$ и разделены пробелами.

**Выходные данные**

Выходной файл OUTPUT.TXT должен содержать одно число – количество деревьев.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 0 0 2 0 0 2 | 6 |
| 2 | 0 0 -1 -1 -1 0 | 3 |

## 359. Змейка - 2

*Время: 1 сек. Память: 16 Мб Сложность: 38%)*

Мальчик Вася на уроке математики, вместо того, чтобы слушать учителя, рисовал числа в тетрадке в клеточку. Да не просто так рисовал, а определенным образом. Сначала он поставил в клетку число 1. Затем справа от нее нарисовал число 2. Затем снизу от числа 2 написал число 3. Затем перешёл на клетку правее и продолжил увлекательное занятие двигаясь по столбцу вверх, пока число в этом столбце не стало выше самого верхнего числа в предыдущем столбце. Затем он перешёл на клетку правее и опять таки продолжил рисование чисел, начиная с 7, но только уже сверху вниз, пока не нарисовал число, которое оказалось на одну клетку ниже самого нижнего числа в предыдущем столбце. И так далее. Вася не любил числа, заканчивающиеся нулем, и пропускал их при рисовании змейки. Первые его шесть заполненных столбцов мы скопировали из его тетрадки и привели здесь на рисунке. Так как Вася очень любопытный, то он очень хочет узнать, какое же число будет у него стоять в N-ом столбце в той строке, где стоит число 1. Первые 6 таких чисел в этой строке видны на рисунке: 1, 2, 5, 8, 14, 19.

Требуется написать программу, которая поможет Васе.

**Входные данные**

Входной файл INPUT.TXT содержит одно число N ($1 \le N \le 10^6$) – номер столбца.

**Выходные данные**

Выходной файл OUTPUT.TXT должен содержать N-ое число в строке, где стоит число 1.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3 | 5 |
| 2 | 5 | 14 |

## 360. Максимальная тройка

*(Время: 1 сек. Память: 16 Мб Сложность: 33%)*

В данной двумерной целочисленной таблице размером N × N требуется найти три элемента, сумма которых максимальна. При этом первый элемент должен быть соседним по горизонтали или вертикали со вторым, а второй – с третьим.

**Входные данные**

Входной файл INPUT.TXT содержит в первой строке число N ($1 \le N \le 2000$). В следующих N строках записано по N чисел – элементы таблицы. Элементы матрицы по абсолютной величине не превышают 100.

**Выходные данные**

Выходной файл OUTPUT.TXT должен содержать единственное число – максимальную сумму.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3<br>1 1 1<br>2 2 1<br>2 1 0 | 6 |

## 361. Подстроки из одинаковых букв

*(Время: 1 сек. Память: 16 Мб Сложность: 52%)*

В заданной строке, состоящей из малых латинских букв, необходимо найти пару самых длинных подстрок, состоящих из одних и тех же букв (возможно, в разном порядке). Например, в строке twotwow это будут подстроки wotwo и otwow.

**Входные данные**

Входной файл INPUT.TXT содержит исходную строку, длиной от 1 до 100 символов.

**Выходные данные**

Выходной файл OUTPUT.TXT должен содержать единственное число – длину подстрок в максимальной паре, или 0, если таких подстрок в строке нет.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | abcde | 0 |
| 2 | abcdea | 5 |

## 362. Открытка и конверт

*(Время: 1 сек. Память: 16 Мб Сложность: 50%)*

Даны размеры прямоугольных открытки и конверта. Требуется определить, поместится ли открытка в конверте.

**Входные данные**

Входной файл INPUT.TXT содержит в первой строке размеры открытки, во второй строке заданы размеры конверта. Все размеры – натуральные числа, не превосходящие 100.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите «Possible», если открытку можно разместить в конверте и «Impossible» в противном случае.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 1 10<br>9 9 | Possible |
| 2 | 10 15<br>15 10 | Possible |
| 3 | 3 4<br>3 3 | Impossible |

## 363. Длинное произведение

*(Время: 1 сек. Память: 16 Мб Сложность: 46%)*

Даны целые неотрицательные числа M и N. Требуется найти произведение этих чисел.

**Входные данные**

Входной файл INPUT.TXT содержит в первой строке число M, а во второй строке – число N ($0 \le M, N \le 102500$) .

**Выходные данные**

В выходной файл OUTPUT.TXT выведите произведение чисел M и N.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 5<br>7 | 35 |
| 2 | 4134937827592<br>784 | 3241791256832128 |
| 3 | 9876543210<br>1023456789 | 101082152001263526 90 |

## 364. Совершенные числа

*(Время: 1 сек. Память: 16 Мб Сложность: 51%)*

Число называется совершенным, если оно равно сумме всех своих делителей, меньших его самого. Требуется найти все совершенные числа от M до N.

**Входные данные**

Входной файл INPUT.TXT содержит числа M и N, разделенные пробелом ($1 \le M \le N \le 5*10^{18}$) .

**Выходные данные**

В выходной файл OUTPUT.TXT выведите по одному числу в строке в порядке возрастания все совершенные числа, находящихся на отрезке [M, N]. В том случае, когда таких чисел нет следует вывести «Absent».

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 6 6 | 6 |
| 2 | 4 5 | Absent |
| 3 | 5 30 | 6<br>28 |

## 365. Разложение на слагаемые

*(Время: 1 сек. Память: 16 Мб Сложность: 54%)*

Требуется вывести все различные представления натурального числа N в виде суммы натуральных чисел. Представления, отличающиеся друг от друга порядком слагаемых, не являются различными.

**Входные данные**

Входной файл INPUT.TXT содержит целое число N ($2 \le N \le 40$).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите все различные представления числа N без повторов в виде суммы по одному на отдельной строке. Как слагаемые, так и сами суммы могут следовать в произвольном порядке.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 4 | 1+1+1+1<br>1+2+1<br>1+3<br>2+2 |
| 2 | 5 | 1+1+1+1+1<br>1+1+1+2<br>1+1+3<br>1+2+2<br>2+3<br>1+4 |

## 366. Выражение

*(Время: 1 сек. Память: 16 Мб Сложность: 56%)*

Даны N целых чисел $X_1$, $X_2$, …, $X_N$. Требуется расставить между ними знаки «+» и «-» так, чтобы значение получившегося выражения было равно заданному целому S.

**Входные данные**

Входной файл INPUT.TXT в первой строке содержит числа N и S. В следующей строке располагается N чисел, разделенных пробелом. Ограничения: $2 \le N \le 24$, $0 \le X_i \le 5*10^7$, $-10^9 \le S \le 10^9$.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите «No solution», если такой результат получить невозможно, иначе выведите получившееся равенство. Если решение не единственное, выведите любое.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3 10<br>15 25 30 | 15+25-30=10 |
| 2 | 2 100<br>10 10 | No solution |

## 367. Степень - 2

*(Время: 1 сек. Память: 16 Мб Сложность: 45%)*

Для натуральных чисел A и B требуется вычислить значение $A^B$.

**Входные данные**

Входной файл INPUT.TXT в первой строке содержит числа A и B, разделенные пробелом ($1 \le A \le 9$, $1 \le B \le 10^4$) .

**Выходные данные**

В выходной файл OUTPUT.TXT выведите одно число – результат возведения в степень, без лидирующих нулей.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3 20 | 3486784401 |
| 2 | 2 16 | 65536 |
| 3 | 5 50 | 888178419700125232338905334472656 25 |

## 368. Маршрут

*(Время: 1 сек. Память: 16 Мб Сложность: 38%)*

В таблице из N строк и N столбцов клетки заполнены цифрами от 0 до 9. Требуется найти такой путь из клетки (1, 1) в клетку (N, N), чтобы сумма цифр в клетках, через которые он пролегает, была минимальной; из любой клетки ходить можно только вниз или вправо.

**Входные данные**

В первой строке входного файла INPUT.TXT находится число N. В следующих N строках содержатся по N цифр без пробелов ($2 \le N \le 250$) .

**Выходные данные**

В выходной файл OUTPUT.TXT выведите N строк по N символов. Символ «#» (решетка) показывает, что маршрут проходит через эту клетку, а «.» (минус) – что не проходит. Если путей с минимальной суммой цифр несколько, можно вывести любой.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3<br>943<br>216<br>091 | #..<br>###<br>..# |

## 369. Гангстеры

*(Время: 1 сек. Память: 16 Мб Сложность: 44%)*

N гангстеров собираются в ресторан. i-й гангстер приходит в момент времени Ti и имеет богатство Pi. Дверь ресторана имеет K+1 степень открытости, они обозначаются целыми числами из интервала [0, K]. Степень открытости двери может изменяться на единицу в единицу времени, то есть дверь может открыться на единицу, закрыться на единицу или остаться в том же состоянии. В начальный момент времени дверь закрыта (степень открытости 0). i-й гангстер заходит в ресторан, только если дверь открыта специально для него, то есть когда степень открытости двери соответствует его полноте Si. Если в момент, когда гангстер подходит к ресторану, степень открытости двери не соответствует его полноте, он уходит и больше не возвращается. Ресторан работает в интервале времени [0, T].

Требуется собрать гангстеров с максимальным суммарным богатством в ресторане, открывая и закрывая дверь соответствующим образом.

**Входные данные**

В первой строке входного файла INPUT.TXT находятся числа N, K, T, во второй – $T_1$, $T_2$, ..., $T_N$, в третьей – $P_1$, $P_2$, ..., $P_N$. в четвёртой – $S_1$, $S_2$, ..., $S_N$. Числа в строках разделены пробелами.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите одно число – максимальное суммарное богатство гангстеров, попавших в ресторан. Если зайти не удалось никому, вывести 0.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 4 10 20<br>10 16 8 16<br>10 11 15 1<br>10 7 1 8 | 26 |
| 2 | 2 17 100<br>5 0<br>50 33<br>6 1 | 0 |

## 370. Площадь многоугольника

*(Время: 1 сек. Память: 16 Мб Сложность: 48%)*

Многоугольник на плоскости задан целочисленными координатами своих N вершин в декартовой системе координат. Требуется найти площадь многоугольника. Стороны многоугольника не соприкасаются (за исключением соседних – в вершинах) и не пересекаются.

**Входные данные**

В первой строке входного файла INPUT.TXT находится число N. В следующих N строках находятся пары чисел (Xi, Yi) – координаты точек. Если соединить точки в данном порядке, а также первую и последнюю точки, получится заданный многоугольник ($3 \le N \le 50\,000$, $-20\,000 \le Xi, Yi \le 20\,000$).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите одно число – площадь многоугольника. Его следует округлить до ближайшего числа с одной цифрой после запятой.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 4<br>5 0<br>0 5<br>-5 0<br>0 -5 | 50.0 |

| | | |
|---|---|---|
| 2 | 4<br>0 4<br>0 0<br>3 0<br>1 1 | 3.5 |

## 371. Дружественные числа

*(Время: 1 сек. Память: 16 Мб Сложность: 58%)*

Два различных натуральных числа называются дружественными, если первое из них равно сумме делителей второго числа, за исключением самого второго числа, а второе равно сумме делителей первого числа, за исключением самого первого числа. Требуется найти все пары дружественных чисел, оба из которых принадлежат промежутку от M до N.

**Входные данные**

Входной файл INPUT.TXT в первой строке содержит натуральные числа M и N, разделенные пробелом, не превышающие $10^6$.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите в каждой строке по паре чисел через пробел. Первое число пары должно быть меньше второго. Строки должны быть отсортированы в порядке возрастания первого числа пары. Если пар дружественных чисел в промежутке нет, то следует вывести «Absent».

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 200 300 | 220 284 |
| 2 | 200 250 | Absent |
| 3 | 185000 205000 | 185368 203432<br>196724 202444 |

**Комментарий к примеру №1**

220=1+2+4+71+142 (все делители числа 284)
284=1+2+4+5+10+11+20+22+44+55+110 (все делители числа 220)

## 372. Скобки - 2

*(Время: 1 сек. Память: 16 Мб Сложность: 52%)*

Вывести все правильные скобочные выражения длиной N, состоящие из круглых и квадратных скобок.

**Входные данные**

Входной файл INPUT.TXT содержит единственное четное натуральное число N, не превышающее 14.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите всевозможные правильные скобочные выражения по одному в каждой строке.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 2 | ()<br>[] |
| 2 | 4 | ()()<br>()[]<br>(())<br>([])<br>[]()<br>[][]<br>[()]<br>[[]] |

## 373. Маршрут - 2

*(Время: 1 сек. Память: 16 Мб Сложность: 47%)*

Дана матрица NxN, заполненная положительными числами. Путь по матрице начинается в левом верхнем углу. За один ход можно пройти в соседнюю по вертикали или горизонтали клетку (если она существует). Нельзя ходить по диагонали, нельзя оставаться на месте.

Требуется найти максимальную сумму чисел, стоящих в клетках по пути длиной K (клетку можно посещать несколько раз).

**Входные данные**

Входной файл INPUT.TXT в первой строке содержит разделенные пробелом числа N и K. Затем идут N строк по N чисел в каждой – данные таблицы. Элементы матрицы – целые числа от 1 до 9999, $2 \le N \le 100$, $1 \le K \le 2000$.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите одно число – максимальную сумму.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 5 7<br>1 1 1 1 1<br>1 1 3 1 9<br>1 1 6 1 1<br>1 1 3 1 1<br>1 1 1 1 1 | 21 |

## 374. Выпуклая оболочка - 2

*(Время: 1 сек. Память: 16 Мб Сложность: 55%)*

На плоскости заданы N точек своими декартовыми координатами. Найти минимальный периметр многоугольника, содержащего все эти точки. Гарантируется, что искомый многоугольник имеет ненулевую площадь.

**Входные данные**

Входной файл INPUT.TXT в первой строке содержит число N, далее – N строк с парами координат ($x_i$, $y_i$). Ограничения: $3 \le N \le 1000$, $-10\,000 \le x_i, y_i \le 10\,000$, все числа целые, все точки различны.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите одно вещественное число – длину периметра полученного многоугольника с точностью не менее $10^{-2}$.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 5<br>1 0<br>0 1<br>-1 0<br>0 -1<br>0 0 | 5.66 |

## 375. Системы счисления

*(Время: 1 сек. Память: 16 Мб Сложность: 53%)*

Дано целое неотрицательное число в m-ричной системе счисления. Требуется вывести это число в k-ричной системе счисления.

**Входные данные**

Входной файл INPUT.TXT в первой строке содержит два числа m и k (в десятичной системе счисления), во второй строке – число для перевода. Ограничения: $2 \le m, k \le 36$, для представления цифр 10...35 используются прописные латинские буквы A...Z соответственно, число разрядов исходного числа не превышает 1000.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите искомое число без лидирующих нулей.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 10 36<br>29234652 | HELLO |
| 2 | 2 10<br>1101 | 13 |

## 376. День рождения - 2

*(Время: 1 сек. Память: 16 Мб Сложность: 32%)*

Заданы день и месяц рождения, а также текущие день, месяц и год. Определить, сколько дней осталось до дня рождения.

Примечание. Високосные годы – это те, номер которых делится на 400, а также те, номер которых делится на 4, но не делится на 100.

**Входные данные**

В первой строке входного файла INPUT.TXT находятся, разделённые пробелами, день и месяц рождения, во второй – разделённые пробелами текущие день, месяц и год. Ограничения: год от 1920 до 3000, месяц от 1 до 12, день от 1 до числа дней в месяце.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите число дней, оставшихся до дня рождения.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 19 04<br>19 04 2002 | 0 |
| 2 | 05 05<br>19 04 2002 | 16 |
| 3 | 29 02<br>28 02 2001 | 1096 |

## 377. Закраска прямой

*(Время: 1 сек. Память: 16 Мб Сложность: 48%)*

На числовой прямой окрасили N отрезков. Известны координаты левого и правого концов каждого отрезка (Li и Ri). Найти длину окрашенной части числовой прямой.

**Входные данные**

Входной файл INPUT.TXT в первой строке содержит число N, в следующих N строках – пары Li и Ri. Ограничения: все числа целые, не превышающие $10^9$ по абсолютной величине, $1 \le N \le 15\,000$.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите длину окрашенной части прямой.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 2<br>1 3<br>2 4 | 3 |
| 2 | 1<br>10 10 | 0 |

## 378. Суммы

*(Время: 1 сек. Память: 16 Мб Сложность: 62%)*

Дано N целых чисел $A_1$, $A_2$, ..., $A_N$. Требуется найти количество различных значений сумм вида $k_1A_1 + k_2A_2 + ... + k_NA_N$.

**Входные данные**

Входной файл INPUT.TXT в первой строке содержит число N, во второй – $A_1$, $A_2$, ..., $A_N$ через пробел. Ограничения: все числа целые, $1 \le N \le 500$, $0 \le A_i \le 100$, $0 \le k_i \le 1$.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите количество различных значений сумм.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3<br>1 1 2 | 5 |
| 2 | 3<br>1 3 2 | 7 |
| 3 | 5<br>49 100 98 49 0 | 10 |

## 379. Игра с датой

*(Время: 1 сек. Память: 16 Мб Сложность: 54%)*

Играют двое. Задаётся какая-то дата 2008 года. Каждый игрок на своём ходе называет более позднюю дату, увеличивая на 1 или 2 либо день в месяце, либо месяц, но не то и другое сразу. При этом сочетание дня и месяца должно оставаться датой. Игрок, назвавший 31 декабря, проигрывает. Оба играют наилучшим образом. Исходя из заданной даты вывести, кто выиграет.

**Входные данные**

В первой строке входного файла INPUT.TXT находятся числа, обозначающие день и месяц. Месяц указывается от 1 до 12, день от 1 до числа дней в месяце, дата «31 декабря» отсутствует во входных данных.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите 1, если выигрывает первый (начинающий) игрок, или 2 – в противном случае.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 30 12 | 2 |
| 2 | 29 12 | 1 |
| 3 | 29 11 | 2 |

## 380. Площадь прямоугольников

*(Время: 1 сек. Память: 16 Мб Сложность: 50%)*

Дано N прямоугольников со сторонами, параллельными осям координат. Требуется определить площадь фигуры, образованной объединением данных прямоугольников.

**Входные данные**

В первой строке входного файла INPUT.TXT находится число прямоугольников – N. Затем идут N строк ($1 \le N \le 100$), содержащих по 4 числа: x1, y1, x2, y2 – координаты двух противоположных углов прямоугольника. Все координаты – целые числа, не превосходящие по абсолютной величине 10 000.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите одно целое число – площадь фигуры.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 1<br>1 1 7 7 | 36 |
| 2 | 2<br>1 1 3 3<br>2 2 4 4 | 7 |

## 381. Lines

*(Время: 1 сек. Память: 16 Мб Сложность: 44%)*

В таблице из N строк и N столбцов некоторые клетки заняты шариками, другие свободны. Выбран шарик, который нужно переместить, и место, куда его нужно переместить. Выбранный шарик за один шаг перемещается в соседнюю по горизонтали или вертикали свободную клетку. Требуется выяснить, возможно ли переместить шарик из начальной клетки в заданную, и, если возможно, то найти путь из наименьшего количества шагов.

**Входные данные**

В первой строке входного файла INPUT.TXT находится число N ($2 \le N \le 50$), в следующих N строках – по N символов. Символом точки обозначена свободная клетка, латинской заглавной O – шарик, @ – исходное положение шарика, который должен двигаться, латинской заглавной X – конечное положение шарика.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите в первой строке Yes, если движение возможно, или No, если нет. Если движение возможно, то далее следует вывести N строк по N символов – как и на вводе, но буква X, а также все точки по пути следует заменить плюсами.

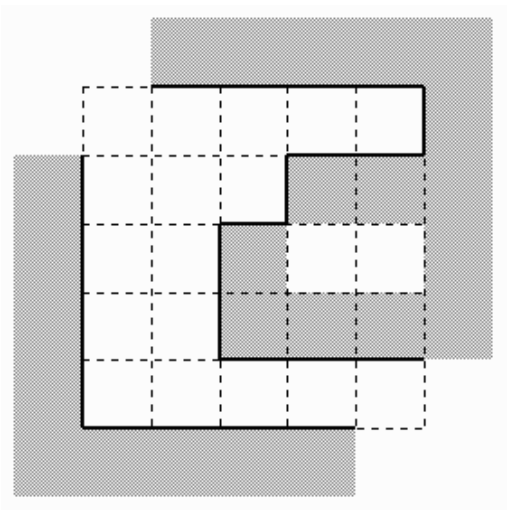
**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 5<br>....X<br>.OOOO<br>.....<br>OOOO.<br>@.... | Yes<br>+++++<br>+OOOO<br>+++++<br>OOOO+<br>@++++ |
| 2 | 5<br>..X..<br>.....<br>OOOOO<br>.....<br>..@.. | No |

## 382. Покраска лабиринта

*(Время: 1 сек. Память: 16 Мб Сложность: 46%)*

Лабиринт представляет собой квадрат, состоящий из NxN сегментов. Каждый из сегментов может быть либо пустым, либо заполненным монолитной каменной стеной. Гарантируется, что левый верхний и правый нижний сегменты пусты. Лабиринт обнесён сверху, снизу, слева и справа стенами, оставляющими свободными только левый верхний и правый нижний углы. Директор лабиринта решил покрасить стены лабиринта, видимые изнутри (см. рисунок). Помогите ему рассчитать количество краски, необходимой для этого.



**Входные данные**

В первой строке входного файла INPUT.TXT находится число N ($3 \le N \le 50$), затем идут N строк по N символов: точка обозначает пустой сегмент, решётка – сегмент со стеной. Размер сегментов – 5x5 метров, высота стен – 5 метров.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите одно число – площадь видимой части внутренних стен лабиринта в квадратных метрах.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 5<br>.....<br>...##<br>..#..<br>..###<br>..... | 550 |

## 383. Красивые числа - 2

*(Время: 1 сек. Память: 16 Мб Сложность: 26%)*

Будем называть число красивым, если сумма его цифр в десятичной системе счисления делится на количество цифр в нем (в десятичной системе счисления).

Необходимо найти N-ое в порядке возрастания красивое число.

**Входные данные**

Входной файл INPUT.TXT содержит целое число N ($1 \le N \le 100\,000$).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите ответ на задачу.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 15 | 20 |

## 384. Числа Фибоначчи - 3

*(Время: 1 сек. Память: 16 Мб Сложность: 52%)*

Последовательностью Фибоначчи называется последовательность чисел $F_0 = 0$, $F_1 = 1$, … , $F_k = F_{k-1} + F_{k-2}$ ($k > 1$).

Требуется найти наибольший общий делитель двух чисел Фибоначчи.

**Входные данные**

Во входном файле INPUT.TXT записаны два целых числа i и j ($1 \le i, j \le 10^6$).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите остаток от деления НОД чисел $F_i$ и $F_j$ на $10^9$.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 5 10 | 5 |
| 2 | 2 4 | 1 |

## 385. Развлечения с измерителем

*(Время: 1 сек. Память: 16 Мб Сложность: 33%)*

Дима обнаружил у папы на столе специальный чертежный прибор, похожий на циркуль-измеритель. Измеритель отличается от обычного циркуля тем, что в обеих его ножках находятся иголки (у обычного циркуля в одной ножке находится иголка, а в другой – грифель).

Кроме измерителя Дима нашел на столе клетчатый лист бумаги, в углах некоторых клеток которого были нарисованы точки. Так как измеритель служит для измерения расстояний, то Дима решил измерить все попарные расстояния между всеми точками на листе бумаги.

Ваша задача – написать программу, которая по координатам точек определит, сколько различных расстояний встречается среди расстояний, которые измерил Дима.

**Входные данные**

Первая строка входного файла INPUT.TXT содержит число n – количество точек ($2 \le n \le 50$). Следующие n строк содержат по два целых числа – координаты точек. Координаты не превышают $10^4$ по абсолютной величине.

**Выходные данные**

На первой строке выходного файла OUTPUT.TXT выведите k – количество различных расстояний, которые измерил Дима. Следующие k строк должны содержать по одному вещественному числу – сами расстояния. Расстояния должны быть выведены в возрастающем порядке. Каждое число должно быть выведено с точностью не менее чем $10^{-9}$.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 4<br>0 0<br>1 1<br>1 0<br>0 1 | 2<br>1.0<br>1.414213562373 |

## 386. Генерация тестов

*(Время: 1 сек. Память: 16 Мб Сложность: 37%)*

При подготовке задач для олимпиад по информатике и программированию часто возникает необходимость подготовки тестов. Поскольку зачастую количество тестов достаточно велико, и каждый из них может содержать множество данных, то генерацию тестов разумно автоматизировать.

В геометрических задачах часто требуется сгенерировать n точек на плоскости так, чтобы никакие три из них не лежали на одной прямой. В этом и состоит ваша задача. Напишите программу, которая по числу N построит множество из N точек, обладающее указанным свойством.

**Входные данные**

Входной файл INPUT.TXT содержит целое число N ($1 \le N \le 300$).

**Выходные данные**

Если искомое множество точек можно построить, то выведите в выходной файл OUTPUT.TXT в первой строке слово YES, а далее N строк, каждая из которых должна содержать два числа – координаты соответствующей точки. Среди точек не должно быть совпадающих. Все координаты должны быть целыми числами, не превосходящими 10000 по абсолютному значению. Если искомое множество точек нельзя построить, выведите в выходной файл строку NO.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 1 | YES<br>0 0 |
| 2 | 4 | YES<br>0 0<br>0 1<br>1 0<br>1 1 |

## 387. Левая рекурсия

*(Время: 1 сек. Память: 16 Мб Сложность: 20%)*

В теории формальных грамматик и автоматов (ТФГиА) важную роль играют так называемые *контекстно-свободные грамматики* (КС-грамматики). КС-грамматикой будем называть четверку, состоящую из множества N нетерминальных символов, множества T терминальных символов, множества P правил (продукций) и начального символа S, принадлежащего множеству N.

Каждая продукция p из P имеет форму A –> *a*, где A нетерминальный символ (A из N), а a – строка, состоящая из терминальных и нетерминальных символов. Процесс вывода слова начинается со строки, содержащей только начальный символ S. После этого на каждом шаге один из нетерминальных символов, входящих в текущую строку, заменяется на правую часть одной из продукций, в которой он является левой частью. Если после такой операции получается строка, содержащая только терминальные символы, что процесс вывода заканчивается.

Во многих теоретических задачах удобно рассматривать так называемые *нормальные формы* грамматик. Процесс приведения грамматики к нормальной форме часто начинается с устранения левой рекурсии. В этой задаче мы будем рассматривать только ее частный случай, называемый непосредственной левой рекурсией. Говорят, что правило вывода A –> R содержит непосредственную левую рекурсию, если первым символом строки R является A.

Задана КС-грамматика. Требуется Найти количество правил, содержащих непосредственную левую рекурсию.

**Входные данные**

Первая строка входного файла INPUT.TXT содержит количество n ($1 \le n \le 1000$) правил в грамматике. Каждая из последующих n строк содержит по одному правилу. Нетерминальные символы обозначаются заглавными буквами латинского алфавита, терминальные - строчными. Левая часть продукции отделяется от правой символами –>. Правая часть продукции имеет длину от 1 до 30 символов.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите ответ на задачу.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3<br>S->Sabc<br>S->A<br>A->AA | 2 |

## 388. Седловые точки

*(Время: 1 сек. Память: 16 Мб Сложность: 28%)*

Задана матрица, содержащая N строк и M столбцов. Седловой точкой этой матрицы назовем элемент, который одновременно является минимумом в своей строке и максимумом в своем столбце.

Найдите количество седловых точек заданной матрицы.

**Входные данные**

Входной файл INPUT.TXT в первой строке содержит целые числа N и M ($1 \le N, M \le 750$). Далее следуют N строк по M чисел в каждой. Элементы матрицы не превосходят 1000 по абсолютной величине.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите ответ на задачу.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 2 2<br>0 0<br>0 0 | 4 |
| 2 | 2 2<br>1 2<br>3 4 | 1 |

## 389. К коду Грея

*(Время: 1 сек. Память: 16 Мб Сложность: 50%)*

Рассмотрим циклическую последовательность попарно различных чисел $\{a_0, a_1, \dots, a_{2^n-1}\}$, $0 \le a_i \le 2^n-1$. Назовем эту последовательность кодом Грея, если любой $a_i$ отличается от левого соседа $a_{i-1}$ и правого соседа $a_{i+1}$ только в одной цифре в двоичной записи этих чисел. Для $a_0$ левым соседом считается $a_{2^n-1}$, а для $a_{2^n-1}$ правым соседом считается $a_0$.

Вася хочет запрограммировать игру-головоломку, которая будет позволять пользователю менять местами два любых числа $a_i$ и $a_j$ . Задача игрока – получить код Грея. Модуль, отвечающий за перестановку чисел, Вася берет на себя. А вот Ваша задача – написать программу, которая будет определять после каждой перестановки – является ли последовательность кодом Грея.

**Входные данные**

В первой строке входного файла INPUT.TXT содержится число n. В следующей строке перечислены попарно различные числа $a_i$. В третьей строке записано число m – количество перестановок, сделанных пользователем. В следующих m строках перечислены числа (i, j) – индексы переставляемых элементов. Ограничения: $1 \le n \le 16$; $1 \le m \le 10^5$; $i \ne j$, $0 \le i, j < 2^n$.

**Выходные данные**

В выходной файл OUTPUT.TXT запишите m строчек – в i-той строке запишите «Yes», если после i-той перестановки последовательность стала кодом Грея и «No» в противном случае.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 20<br>1 3 2<br>2<br>1 2<br>2 1 | No<br>Yes |

## 390. Треугольная область

*(Время: 1 сек. Память: 16 Мб Сложность: 41%)*

Одним из субъектов Флатландии является Треугольная область. Как следует из ее названия, она имеет форму треугольника, вершины которого находятся в точках с координатами $(x_1, y_1)$, $(x_2, y_2)$, $(x_3, y_3)$. Административный центр этой области находится в точке с координатами $(x_c, y_c)$, которая лежит строго внутри указанного треугольника.

Для оценки транспортного и логистического потенциала области, ее руководству понадобилось узнать расстояние от административного центра области до ее границы. Напишите программу, которая вычислит это расстояние.

**Входные данные**

Первая строка входного файла INPUT.TXT содержит шесть целых чисел – $x_1$, $y_1$, $x_2$, $y_2$, $x_3$, $y_3$. Вторая строка входного файла содержит два целых числа – $x_c$ и $y_c$. Все числа во входном файле не превосходят 10000 по абсолютной величине. В заданном треугольнике нет тупых углов.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите искомое расстояние до границы с точностью не менее $10^{-6}$.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 0 0 1 2 2 0<br>1 1 | 0.4472135 |
| 2 | 0 0 0 3 3 0<br>1 1 | 0.7071067 |

## 391. Взлом хеш-функции

*(Время: 1 сек. Память: 16 Мб Сложность: 35%)*

В некоторых задачах защиты информации используются так называемые хеш-функции. Одним из важнейших классов хеш-функций являются так называемые полиномиальные хеш-функции.

Пусть дана строка $S = s_1 s_2 \ldots s_l$, состоящая из цифр от 0 до 9. Тогда значение полиномиальной хеш-функции $p(S, x, m)$ вычисляется следующим образом: $p(S,x,m) = \left( \sum_{i=1}^{l} s_i x^{i-1} \right) \bmod m$, (a mod b обозначает остаток от деления числа a на число b). Например, пусть $S$ = '0123', тогда $p(S, 2, 5) = (0 \cdot 1 + 1 \cdot 2 + 2 \cdot 4 + 3 \cdot 8) \bmod 5 = 4$.

Одним из способов применения хеш-функций является хранение паролей. Часто бывает так, что пароли приходится хранить в незащищенной таблице базы данных, поэтому вместо них хранят хеш-функции от них. При проверке пароля вычисляется хеш-функция от введенной строки и сравнивается со значением, хранящимся в таблице.

Ваша задача состоит в том, чтобы по заданным числам x, m, L и v найти строку S из цифр от 0 до 9 длины L, значение полиномиальной хеш-функции p(S, x, m) которой равно v.

**Входные данные**

Входной файл INPUT.TXT содержит четыре целых числа: x (x – простое число, $5 \le x \le 100$), m (m является степенью двойки, $1 \le m \le 256$), L ($10 \le L \le 100$) и v ($0 \le v \le m-1$).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите искомую строку или NO SOLUTION, если такой строки не существует.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 5 16 10 9 | 0422207956 |

## 392. Сдвиг перестановки

*(Время: 1 сек. Память: 16 Мб Сложность: 24%)*

*Перестановкой* порядка n называется последовательность из попарно различных целых положительных чисел $p_1, p_2, \ldots, p_n$, где каждое $1 \le p_i \le n$. Будем говорить, что перестановка $q_1, q_2, \ldots, q_n$ лексикографически меньше перестановки $p_1, p_2, \ldots, p_n$, если существует такое i, что $q_i < p_i$, а для любого $j < i$ $p_j = q_j$.

*Циклическим сдвигом* на k перестановки $p_1, p_2, \ldots, p_n$ называется последовательность $p_{k+1}, p_{k+2}, \ldots, p_n, p_1, \ldots, p_k$. Отметим, что любой циклический сдвиг перестановки также является перестановкой.

Ваша задача состоит в том, чтобы найти наименьший лексикографически циклический сдвиг заданной перестановки.

**Входные данные**

Первая строка входного файла INPUT.TXT содержит порядок n ($1 \le n \le 10^5$) заданной перестановки. Вторая строка содержит числа $p_1, p_2, \ldots, p_n$, отделенные друг от друга пробелами.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите перестановку, являющуюся наименьшим лексикографически циклическим сдвигом перестановки, заданной во входном файле.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3<br>3 2 1 | 1 3 2 |

## 393. Плейлист

*(Время: 1 сек. Память: 16 Мб Сложность: 48%)*

Антон – большой любитель музыки. Но в разные моменты ему нравится разная музыка. Однажды он решил составить рейтинг своих песен, причем считать его так – если песня дослушана до конца, то ее рейтинг увеличивается на единицу, если же он так и не дослушал ее, переключив на следующую, то уменьшается на единицу. Отметим, что вследствие этого рейтинг может стать отрицательным.

Он попросил Вас написать ему программу, которая подсчитывала бы такой рейтинг. Для этого он записал в каком порядке прослушивались песни и в какие моменты времени он переключал песню на следующую. Изначально рейтинги всех песен равны нулю.

**Входные данные**

Первая строка входного файла INPUT.TXT содержит одно число n ($1 \le n \le 1000$) – количество песен у Антона. В следующих n строках следуют описания песен – название песни, состоящее не более, чем из 50 маленьких латинских букв, и длина песни в секундах, разделенные пробелом. Каждая песня длится положительное число секунд и не более 30 минут. (n + 2)-я строка содержит два числа m, k ($0 \le k \le m \le 1000$) – количество прослушанных песен и количество переключений на следующую. В следующих m строках следуют названия прослушанных песен в порядке их прослушивания (все эти песни из списка). И, наконец, в последней строке – k целых неотрицательных чисел – последовательные (в неубывающем порядке) времена переключения на следующую песню (гарантируется, что она всегда существует), отсчитываемые в секундах от начала прослушивания первой. Считается, что если Антон переключил песню в момент окончания какой-нибудь, то он пропускает уже следующую.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите n чисел – рейтинги песен в том порядке, в котором они даны во входном файле.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 2<br>songa 10<br>songb 20<br>5 2<br>songa<br>songb<br>songb<br>songa<br>songa<br>40 55 | 1 0 |

## 394. Апельсины

*(Время: 1 сек. Память: 16 Мб Сложность: 31%)*

Катя решила пригласить к себе в гости n друзей. Так как ее друзья очень любят фрукты, то в качестве угощения для них она купила m одинаковых апельсинов.

Она хочет разрезать каждый апельсин на одинаковое число равных долек так, чтобы их можно было распределить между гостями (сама Катя апельсины есть не будет), и всем гостям досталось поровну долек.

Напишите программу, которая вычисляет минимальное количество долек, на которое необходимо разрезать каждый апельсин, чтобы были выполнены указанные выше условия.

**Входные данные**

Входной файл INPUT.TXT содержит два положительных целых числа n и m ($1 \le n, m \le 10^9$).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите ответ на задачу.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 2 5 | 2 |
| 2 | 2 4 | 1 |

## 395. Произведение цифр - 2

*(Время: 1 сек. Память: 16 Мб Сложность: 28%)*

Найдите количество чисел из отрезка [l, r], которые делятся на произведение своих цифр.

**Входные данные**

Входной файл INPUT.TXT содержит два целых числа l и r ($1 \le l \le r \le 10^9$, $|r-l| \le 10^5$) .

**Выходные данные**

В выходной файл OUTPUT.TXT выведите ответ на задачу.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 1 12 | 11 |

## 396. Точки и отрезки

*(Время: 2 сек. Память: 16 Мб Сложность: 62%)*

Дано N отрезков на числовой прямой и M точек на этой же прямой. Для каждой из данных точек определите, скольким отрезкам она принадлежит. Точка x считается принадлежащей отрезку с концами a и b, если выполняется двойное неравенство $\min(a, b) \le x \le \max(a, b)$.

**Входные данные**

Первая строка входного файла INPUT.TXT содержит два целых числа N – число отрезков и M – число точек ($1 \le N, M \le 10^5$). В следующих N строках по два целых числа $a_i$ и $b_i$ – координаты концов соответствующего отрезка. В последней строке M целых чисел – координаты точек. Все числа во входном файле не превосходят по модулю $10^9$.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите M чисел – для каждой точки количество отрезков, в которых она содержится.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3 2<br>0 5<br>-3 2<br>7 10<br>1 6 | 2 0 |
| 2 | 1 3<br>-10 10<br>-100 100 0 | 0 0 1 |

## 397. Качество строки

*(Время: 1 сек. Память: 16 Мб Сложность: 52%)*

Назовем *качеством строки* разность между максимальным и минимальным номерами в алфавите букв, входящих в строку. Например, качество строки ab равно 2 - 1 = 1, а строки abcz равно 26 - 1 = 25.

Дана строка S. Необходимо найти непустую подстроку этой строки, обладающую максимальным качеством, а из всех таких – минимальную по длине.

**Входные данные**

Входной файл INPUT.TXT содержит непустую строку S, состоящую из строчных букв латинского алфавита. Ее длина не превосходит $2 \cdot 10^5$ символов.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите искомую подстроку. Если вариантов ответа несколько, выведите любой.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | aba | ab |
| 2 | zzz | z |

## 398. Сумма - 2

*(Время: 1 сек. Память: 16 Мб Сложность: 25%)*

Задано натуральное число число x. Найдите число способов представить его в виде суммы четырех натуральных чисел: $x = a + b + c + d$, где $a \le b \le c \le d$.

**Входные данные**

Входной файл INPUT.TXT содержит целое число x ($1 \le x \le 1500$).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите ответ на задачу.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3 | 0 |
| 2 | 5 | 1 |

## 399. Жук

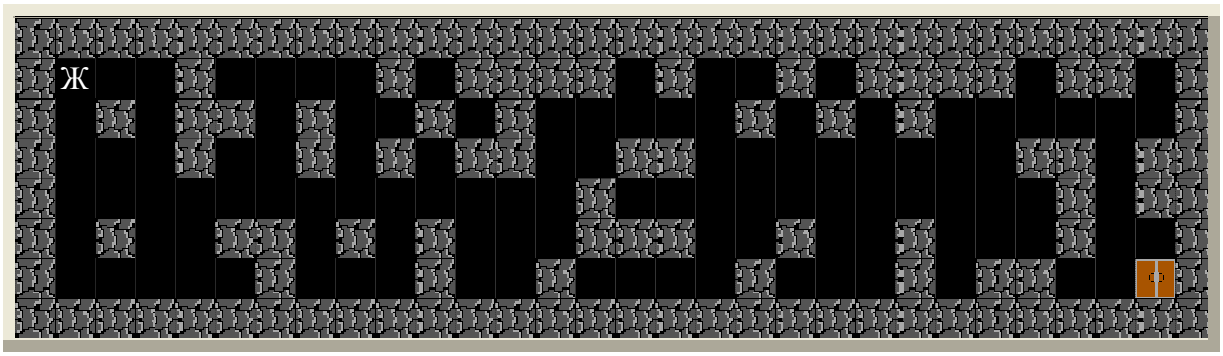
*(Время: 1 сек. Память: 16 Мб Сложность: 30%)*

Петя нашел в Интернете по адресу **http://buglab.ru** игру-головоломку «Жук», в которой от участников требуется построить для жука лабиринт таким образом, чтобы жук как можно дольше искал выход.

Жук всегда начинает свое движение с левого верхнего угла, а выход всегда находится в правом нижнем. Жук движется не оптимально, а следующим образом: он идет туда, где еще не был, либо был там реже. Т.е. проходя каждую клетку лабиринта, жук запоминает: сколько раз он был в этой клетке и при обдумывании направления своего движения в какой то конкретный момент он смотрит: сколько раз он был в клетке снизу, сколько справа, сколько слева и сколько сверху и движется туда, где он был меньше раз. Если таких направлений несколько и одно из них совпадает с текущим направлением движения, то он не меняет направления, иначе он движется согласно следующим приоритетам: вниз, направо, вверх, налево. Т.е. если минимальное число посещений сразу справа и слева (а двигался он при этом вверх или вниз), то жук идет направо, т.к. у «направо» приоритет выше. Следует заметить, что двигаясь по данному алгоритму жук всегда достигнет выхода в том случае, когда выход существует.

Изучив алгоритм движения жука Петя хочет написать программу, которая по заданному лабиринту определит количество перемещений жука прежде, чем он достигнет выхода. Помогите Пете с реализацией данной программы!

**Конструктор лабиринта**




- кнопка запуска жука

**Ходы: 0**

**Входные данные**

Входной файл INPUT.TXT в первой строке содержит разделенные пробелом целые числа N и M – количество строк и столбцов в лабиринте ($4 \le N, M \le 100$). Далее следует N строк, содержащих данные лабиринта построчно. Каждая строка содержит M символов – клетки лабиринта текущей строки, где символ «@» обозначает присутствие стены, а символ пробела – пустое пространство. Гарантируется, что граница лабиринта окружена стеной. Предполагается, что жук начинает свое движение из координаты (2, 2) и заканчивает в координате (M-1, N-1), подразумевается, что в этих координатах нет стен.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите количество движений жука, если спасительный маршрут для жука существует, и -1 в противном случае.

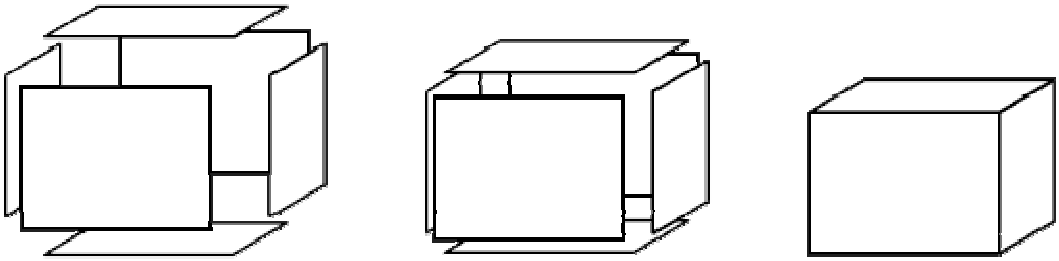
**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 6 6<br>@@@@@@<br>@ @<br>@ @<br>@ @ @@<br>@ @ @<br>@@@@@@ | 20 |
| 2 | 8 30<br>@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@<br>@ @ @ @@@@ @ @ @@@@ @@ @<br>@ @ @@ @ @ @ @ @ @ @<br>@ @ @ @ @@ @@ @@ @@<br>@ @ @ @ @@<br>@ @@ @ @ @ @ @ @ @<br>@ @ @ @ @ @ @ @@ @<br>@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ | 630 |
| 3 | 4 4<br>@@@@<br>@ @@<br>@@ @<br>@@@@ | -1 |

## 400. Коробка

*(Время: 1 сек. Память: 16 Мб Сложность: 50%)*

Иван работает на заводе, который производит тяжелую технику. Его работа очень проста – он собирает коробки и упаковывает в них технику для заказчиков. Каждая такая коробка представляет собой параллелепипед. Для сборки коробки Иван использует шесть прямоугольных деревянных плиток. Каждая плита представляет собой одну из сторон коробки.



Петр подбирает плитки для Ивана. Петр недостаточно умен и поэтому часто допускает ошибки – он приносит Ивану такие плитки, из которых невозможно собрать коробку. Но Иван не доверяет Петру. Поэтому он всегда тратит массу времени на то, чтобы объяснить Петру то, где он допустил ошибку.

К счастью, Петр обожает все, что связано с компьютерами и верит в то, что компьютеры никогда не ошибаются. Иван решил, что можно использовать это в их работе. Иван попросил Вас написать программу, которая по заданным размерам шести плиток скажет: возможно ли построить из них коробку.

**Входные данные**

Входной файл INPUT.TXT содержит шесть строк, каждая из которых содержит два натуральных числа w и h ($1 \le w, h \le 10\,000$) – ширина и высота плиты в миллиметрах.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите «POSSIBLE», если возможно собрать коробку из данных плит, и «IMPOSSIBLE» в противном случае.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 1345 2584<br>2584 683<br>2584 1345<br>683 1345<br>683 1345<br>2584 683 | POSSIBLE |
| 2 | 1234 4567<br>1234 4567<br>4567 4321<br>4322 4567<br>4321 1234<br>4321 1234 | IMPOSSIBLE |

## 401. Шары и коробки

*(Время: 1 сек. Память: 16 Мб Сложность: 52%)*

У вас имеется N выстроенных в ряд коробок, A красных и B синих шаров. Все красные шары (аналогично и синие) идентичны. Вы можете класть шары в коробки. Разрешается размещать в коробках шары как одного, так и двух видов одновременно. Так же разрешается оставлять некоторые из коробок пустыми. Не обязательно класть все шары в коробки.

Требуется написать программу, которая определяет количество различных способов, которыми возможно заполнить коробки шарами.

**Входные данные**

Входной файл INPUT.TXT содержит целые числа N, A, B ($1 \le N \le 20$, $0 \le A, B \le 20$).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите ответ на задачу.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 1 1 1 | 4 |
| 2 | 2 1 1 | 9 |

## 402. Коттеджный поселок

*(Время: 3 сек. Память: 16 Мб Сложность: 75%)*

Поблизости от столицы Флатландии одна компания решила построить коттеджный поселок. Строительная компания, которая занимается возведением коттеджей, решила раскрасить некоторые коттеджи в розовый цвет, а остальные – в голубой. Но они не могут решить, какой коттедж раскрасить в какой цвет.

Директор компании утверждает, что раскраска симпатичная, если есть хотя бы один розовый коттедж, хотя бы один голубой коттедж, и можно провести такую прямую, что все розовые коттеджи окажутся с одной стороны от нее, а все голубые – с другой стороны (при этом на самой прямой коттеджей быть не должно). На это главный дизайнер возразил, что есть несколько способов сделать симпатичную раскраску.

Помогите им определить, сколько существует различных симпатичных раскрасок.

**Входные данные**

Первая строка входного файла INPUT.TXT содержит число N – количество коттеджей ($1 \le N \le 300$). Следующие N строк содержат координаты коттеджей, каждая строка содержит два целых числа $x_i$ и $y_i$ ($-10^4 \le x_i, y_i \le 10^4$).

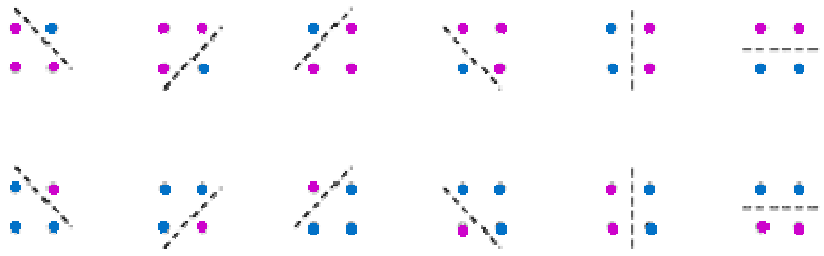
**Выходные данные**

В выходной файл OUTPUT.TXT выведите одно число – ответ на задачу.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 4<br>0 0<br>1 0<br>1 1<br>0 1 | 12 |

**Пояснение к примеру**



## 403. Обмен валюты

*(Время: 1 сек. Память: 16 Мб Сложность: 70%)*

Петя работает в обменном пункте во Флатландии. Недавно Петя получил от начальства набор цифр для отображения обменного курса. К сожалению, набор содержит всего по две копии каждой цифры. Теперь Петя хочет узнать, сколько различных обменных курсов он сможет отобразить.

Петя обменивает флатландские доллары на крайландские тугрики. Петя уверен, что курс обмена будет целым числом, которое находится в диапазоне от L до R, включительно.

**Входные данные**

Входной файл INPUT.TXT содержит два целых числа L и R ($1 \le L \le R \le 10^{18}$).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите одно целое число – количество обменных курсов, которые Петя может отобразить с использованием полученного набора.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 1 1000 | 990 |
| 2 | 1 100 | 100 |

## 404. Игра с камнями

*(Время: 2 сек. Память: 16 Мб Сложность: 63%)*

Правила игры: исходно на столе располагается кучка, содержащая N камней. Два игрока по очереди берут камни из кучки. Если перед ходом игрока кучка содержит K камней, игрок может взять из нее от 1 до [sqrt(K)] (целую часть от квадратного корня из K) камней, включительно. Например, если в кучке 10 камней, разрешается взять 1, 2 или 3 камня. Если в кучке не осталось камней, то игрок, который должен сделать ход, проигрывает.

Требуется определить: сможет ли первый игрок выиграть, если оба игрока будут придерживаться оптимальной стратегии.

**Входные данные**

Входной файл INPUT.TXT содержит натуральное число N – число камней в кучке ($N \le 10^{12}$).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите WIN, если первый игрок может выиграть вне зависимости от действий второго игрока, либо LOSE, если второй игрок может заставить первого игрока проиграть.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3 | WIN |
| 2 | 5 | LOSE |

## 405. Туристическое агентство

*(Время: 5 сек. Память: 32 Мб Сложность: 80%)*

Антон работает в межгалактическом туристическом агентстве. Довольно часто ему приходится прокладывать путь с одной планеты на другую с использованием существующих рейсов космических кораблей. К сожалению, количество рейсов невелико, поэтому пассажирам часто приходится пересаживаться на промежуточных планетах.

Антон заметил, что некоторые планеты используются в качестве промежуточных чаще, чем другие. Он решил провести исследование – для каждой планеты A он хотел бы узнать, сколько существует пар различных планет (B,C), таких что любой путь с планеты B на планету C проходит через планету A.

Помогите Антону!

**Входные данные**

Первая строка входного файла INPUT.TXT содержит два целых числа: N и M – количество планет и количество рейсов космических кораблей, соответственно ($2 \le N \le 20\,000$, $1 \le M \le 200\,000$). Следующие M строк описывают рейсы космических кораблей. Каждый рейс связывает две планеты, и им можно воспользоваться в любом из двух направлений. С любой планеты можно добраться до любой другой.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите N целых чисел – для каждой планеты A выведите количество пар различных планет, таких, что любой путь с одной планеты на другую проходит через A.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 7 9<br>1 2<br>1 3<br>1 4<br>1 5<br>1 6<br>1 7<br>2 3<br>4 5<br>6 7 | 18<br>6<br>6<br>6<br>6<br>6<br>6 |

## 406. Криптография

*(Время: 1 сек. Память: 16 Мб Сложность: 38%)*

Подстановочным называется шифр, при использовании которого каждая буква сообщения заменяется какой-нибудь другой буквой, а все пробелы и знаки препинания удаляются. При этом для сохранения возможности однозначной расшифровки никакие две буквы не превращаются в одну.

Зная тот факт, что частоты появления в осмысленном тексте букв различаются, можно с хорошей точностью установить, каким образом были зашифрованы буквы. Среди всех возможных способов перестановки букв следует выбрать такой, для которого минимально значение величины

$$\sum_c |e_c - p_c|.$$

Здесь через $e_c$ и $p_c$ обозначены частоты символа c, рассчитанные для среднего текста и для записки, получающейся из зашифрованного текста после применения данной перестановки.

Требуется по заданным частотам написать программу, расшифровывающую закодированное послание.

**Входные данные**

В первой строке входного файла INPUT.TXT записаны два целых числа – N и M ($1 \le N \le 26$, $1 \le M \le 10^5$). Следующие N строк задают буквы, которые, по сведениям Кристины, могли встречаться в исходном тексте. Соответствующая буква ci – первый символ строки, через пробел от него записана частота этой буквы pi, заданная с 4 знаками после десятичной точки. Все pi неотрицательны, а их сумма равна 1. Последняя строка содержит m символов – текст, который необходимо расшифровать.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите N символов по одному на строке, которые, если верить описанному методу, должны быть сопоставлены символом из входного файла. Если оптимальных решений несколько, выведите любое из них.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3 5<br>t 0.1667<br>r 0.5000<br>q 0.3333<br>ecere | r<br>e<br>c |

## 407. Сдача

*(Время: 1 сек. Память: 16 Мб Сложность: 44%)*

Когда Миша и Маша покупали подарок, возникла интересная ситуация. У них была в распоряжении только одна большая купюра, а у продавца – некоторое количество мелочи. Дело происходило утром, поэтому продавцу нужно было экономить мелочь, и он хотел отдать сдачу минимальным количеством монет. Подумав некоторое время, они точно определили, с каким количеством монет продавцу придется расстаться.

А вы сможете решить такую задачу?

**Входные данные**

В первой строке входного файла INPUT.TXT записано число N ($1 \le N \le 10$) – количество различных номиналов монет, содержащихся в кассе. Можно считать, что количество монет каждого номинала достаточно. На следующей строке содержится N целых чисел $a_i$ ($0 < a_i \le 2000$) – номиналы монет. В третьей строке записано одно число K ($1 \le K \le 10^6$) – сумма, которую нужно набрать.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите минимальное количество монет, которое придется отдать продавцу, или -1, если продавец вообще не сможет дать им сдачу.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3<br>1 5 7<br>19 | 3 |
| 2 | 5<br>7 8 9 10 11<br>13 | -1 |

## 408. Письмо

*(Время: 1 сек. Память: 16 Мб Сложность: 31%)*

Вася хочет отправить Пете письмо на листе шириной K. Он хочет распечатать на нем текст, состоящий из N строк. Необходимо изменить количество ведущих и концевых пробелов так, чтобы строки оказались посередине листа, и их длина стала равна K. Вася считает, что строка находится посередине листа, если количество ведущих пробелов не превосходит количества концевых пробелов и, если при сдвиге строки на один символ вправо (т.е. при добавлении одного пробела в начало строки), указанное свойство нарушается.

**Входные данные**

Первая строка входного файла INPUT.TXT содержит два целых числа K и N ($1 \le K \le 100$, $1 \le N \le 1000$). Следующие N строк содержат текст Васиного письма. Каждая строка письма содержит хотя бы один символ, отличный от пробела. Длина каждой строки во входном файле не превосходит 100.

**Выходные данные**

Если Вася сможет написать письмо, удовлетворяющее всем его требованиям, то в выходной файл OUTPUT.TXT выведите отформатированный текст письма, иначе, выведите фразу «Impossible.» (без кавычек).

**Примеры** (В примерах для большей наглядности пробелы заменены на плюсы!)

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 20 3<br>++Привет!++<br>+Напиши+мне.++<br>++++Пока+=)+ | ++++++Привет!+++++++<br>++++Напиши+мне.+++++<br>++++++Пока+=)+++++++ |
| 2 | 5 1<br>Привет. | Impossible. |

## 409. Железная дорога

*(Время: 1 сек. Память: 16 Мб Сложность: 26%)*

При строительстве новой железной дороги возникли проблемы. Дорога пролегает по холмистой местности, однако сами пути должны идти строго горизонтально. Поэтому руководство строительной компании приняло решение выровнять поверхность земли на этом участке. Главная проблема состоит в том, что привозить или вывозить землю на стройку стоит 10000$ за кубический метр. Поскольку бюджет железной дороги невелик, этого нельзя себе позволить.

Поэтому главный инженер принял решение выровнять поверхность, используя только землю, из которой состоят холмы. Теперь самая сложная задача состоит в том, чтобы выяснить высоту над уровнем моря, на которой будет пролегать дорога. Это ответственное задание было поручено Вам.

Через каждый метр от начала участка была измерена высота над уровнем моря. Напишите программу, которая по данным измерений рассчитывает искомую высоту.

**Входные данные**

Первая строка входного файла INPUT.TXT содержит количество N ($1 < N \le 30000$) точек, в которых была замерена высота. Вторая строка содержит результаты замеров – i-ое число строки содержит высоту над уровнем моря точки, находящейся на расстоянии (i-1) метр от начала участка. Все высоты – целые неотрицательные числа, не превосходящие 10000. Считайте, что между соседними точками измерений земная поверхность строго прямолинейна.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите ответ на задачу с точностью, не меньшей $10^{-5}$.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 4<br>0 1 1 0 | 0.6666666667 |
| 2 | 5<br>2 2 2 2 2 | 2.0000000000 |

## 410. Цифровое колдовство

*(Время: 1 сек. Память: 16 Мб Сложность: 75%)*

В современном мире все становится цифровым. Шаманы в одной сказочной северной стране тоже стараются не отставать от жизни. Последние изыскания показали, что в качестве материальных компонентов для колдовства с большой эффективностью можно использовать длинные веревки, сплетенные из моржовых усов. Каждая веревка кодирует магическое целое число следующим образом: сначала она определенным образом делится на части, соответствующие цифрам магического числа, в каждой из которых можно завязать от нуля до девяти узелков (глобализация вынуждает шаманов использовать обычную десятичную систему счисления). Конечно, при этом для задания очень больших чисел могут понадобиться длинные веревки. Можно надеяться, что вводная часть вам ясна. Шаман племени X наслал порчу на племя Y.

Шаману племени Y удалось выяснить, какое магическое число N было использовано при проведении страшного ритуала. Теперь ему нужно создать веревку с таким числом K, чтобы N + K делилось на фундаментальную магическую константу M. Необходимо сделать это как можно быстрее, поэтому он хочет выполнить задачу, завязав наименьшее число узелков, даже если для этого придется взять самую длинную веревку из потайного хранилища.

Поможете ли вы спасти племя Y (а вслед за ним, может быть, и весь мир) от ужасного шамана племени X?

**Входные данные**

Во входном файле INPUT.TXT записаны два целых числа – N и M ($0 \le N \le 1000$, $1 < M \le 1000$).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите целое неотрицательное число K – ответ на задачу.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 5 8 | 11 |
| 2 | 6 15 | 24 |

## 411. Квадратное уравнение

*(Время: 1 сек. Память: 16 Мб Сложность: 28%)*

Сложно найти человека, который любит решать однообразные задачки по математике. В последнее время школьникам стало легче, ведь с появлением компьютеров почти в каждой квартире стало существенно проще проверять себя.

Но программы, в которых решение уравнений является стандартной функцией, установлены не везде. Напишите программу, которая сможет решить уравнение

$$ax^2 + bx + c = 0$$

при заданных коэффициентах a, b и c.

**Входные данные**

Единственная строка входного файла INPUT.TXT содержит три целых числа a, b и c, каждое из которых не превосходит по модулю 30000. Числа разделяются пробелами.

**Выходные данные**

На первой строке выходного файла OUTPUT.TXT выведите число корней заданного уравнения. Затем выведите сами корни по одному на строке с ошибкой, не превосходящей $10^{-4}$. Если для заданных коэффициентов корней бесконечно много, на единственной строке выходного файла выведите -1.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 1 -3 2 | 2<br>1.000000<br>2.000000 |
| 2 | 0 -1 6 | 1<br>6.000000 |

## 412. Ферзь и король

*(Время: 1 сек. Память: 16 Мб Сложность: 32%)*

Вася продолжает заниматься шахматами, и теперь он изучает различные окончания. Оказалось, что компьютер очень удобно использовать для анализа позиций. Теперь Вася просит вас написать программу, которая сможет определить, что черный король находится под шахом.

Для начала Васе подойдет программа, которая анализирует игровую ситуацию с тем предположением, что на доске находятся три фигуры – белые король и ферзь и черный король. Черный король находится под шахом, если белый ферзь может за один ход попасть на занимаемую им клетку. Шахматный ферзь может перемещаться по вертикали, горизонтали или диагонали, но, в отличие от коня, не может «перепрыгивать» через другие фигуры.

**Входные данные**

В единственной строке входного файла INPUT.TXT записаны обозначения трех клеток шахматной доски, разделенные пробелами: положения белого короля, белого ферзя и черного короля соответственно. При этом гарантируется, что черный и белый короли не находятся на соседних клетках.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите слово YES, если черный король находится под шахом и NO, если шаха нет.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | d3 d5 d8 | YES |
| 2 | a1 a8 b3 | NO |

## 413. Военная база

*(Время: 1 сек. Память: 16 Мб Сложность: 42%)*

Со спутника-шпиона получено изображение в некотором волновом диапазоне сверхсекретной военной базы предполагаемого противника. База расположена на Антарктиде, все постройки на ней высечены из кубов льда и имеют на фотографии квадратную форму и не имеют общих фрагментов стен ненулевой длины (по всей видимости, это сделано в целях маскировки от локаторов, работающих в инфракрасном спектре). Благодаря мастерству операторов оказалось, что стены разных построек параллельны границам фотографии.

Для того, чтобы составить сверхсрочный отчет для командования, необходимо узнать, сколько зданий находятся на базе. Напишите программу, которая это сделает.

**Входные данные**

В первой строке входного файла INPUT.TXT записаны числа N и M ($1 \le M, N \le 500$) – размеры фотографии в пикселях по вертикали и по горизонтали. Следующие N строк содержат по M символов каждая: символ «.» соответствует пустому месту, «#» – элементу постройки.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите единственное число – количество построек на базе.

**Пример**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 8 6<br>......<br>...##.<br>...##.<br>......<br>.###..<br>.###..<br>.###..<br>...... | 2 |

## 414. Расследование

*(Время: 1 сек. Память: 16 Мб Сложность: 40%)*

В городском управлении милиции одного прибрежного города ведется расследование крупного дела, в котором могут быть замешаны сотрудники милиции. Было принято решение о тайной установке оборудования для просмотра информации, поступающей через Интернет. Под подозрение попадают два отдела, но добиться выделения денег на покупку двух комплектов оборудования не удалось. К счастью, внутренняя сеть управления имеет древовидную структуру, то есть каждый отдел имеет выход в Интернет через какой-либо другой отдел. Исключение составляет отдел по борьбе с компьютерными преступлениями, который имеет непосредственный доступ в Интернет по модемной линии.

Можно было бы установить оборудование для слежения прямо в этом отделе, но для предотвращения злоупотреблений лучше найти такое расположение, чтобы нарушалась секретность как можно меньшего количества лишних отделов.

Как наиболее опытному в подобных вопросах сотруднику, решение этой задачи поручили вам. Подчиненные уже пронумеровали все отделы натуральными числами, начиная с 1, первый номер присвоен отделу по борьбе с компьютерными преступлениями.

**Входные данные**

Первая строка входного файла INPUT.TXT содержит натуральное число N ($N \le 30000$) – количество отделов. Во второй строке записаны номера отделов, за которыми необходимо установить слежение. На третьей строке находятся n-1 натуральных чисел, i-е из них не больше i и задает номер отдела, к которому подсоединен отдел i + 1.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите одно число – номер отдела, в котором следует установить следящее оборудование.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 4<br>3 4<br>1 1 3 | 3 |
| 2 | 8<br>3 6<br>1 1 2 4 5 1 1 | 1 |

## 415. Подпись

*(Время: 1 сек. Память: 16 Мб Сложность: 30%)*

Марсиане Миша и Маша решили вместе подобрать подарок на день рождения Кати. Когда они наконец нашли то, что хотели, и упаковали предмет в красивую коробку, надо было решить, как подписать подарок. Друзья подумали, что лучшим решением будет составить общую подпись так, чтобы в ней как подстроки содержались их имена.

Учтите, что на Марсе принято подписываться полными именами, а они у марсиан могут быть достаточно длинными.

**Входные данные**

Входной файл INPUT.TXT содержит две строки, в которых записаны полные имена друзей. Имена, как ни странно, состоят из букв латинского алфавита, из которых только первая – прописная. Длина имен не превосходит 1000.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите кратчайшую строку, в которой встречаются имена Миши и Маши одновременно. Буквы, с которых имена начинаются в этой строке нужно сделать большими. Если существует несколько решений, выведите то, которое меньше в алфавитном порядке.

**Примеры**

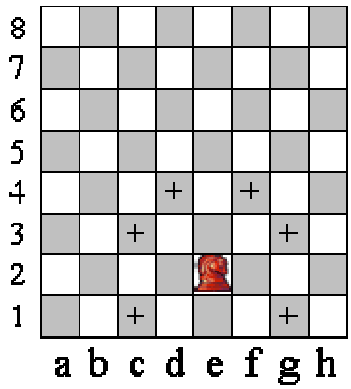
| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | Misha<br>Masha | MashaMisha |
| 2 | Julya<br>Lyalya | JuLyalya |

## 416. Шахматный конь

*(Время: 1 сек. Память: 16 Мб Сложность: 25%)*

Вася решил научиться играть в шахматы. Он нашел книгу с записями партий и внимательно их изучает. Может быть, когда-нибудь Вася станет великим шахматистом, но пока он еще учится в начальной школе, и ему нелегко дается шахматная нотация. Больше всего трудностей у Васи вызывают ходы шахматного коня. Он попросил вас написать программу, которая сможет сообщить Васе, на какие клетки можно пойти конем с заданной клетки.

Вы, наверное, тоже знаете, что конь в шахматах всегда перемещается либо на две клетки по горизонтали и на одну по вертикали, либо на одну по горизонтали и на две по вертикали. Вертикали обозначаются маленькими латинскими буквами от a до h, а горизонтали – цифрами от 1 до 8. Любая клетка на шахматной доске обозначается буквой соответствующей вертикали и цифрой соответствующей горизонтали, например, c6 или e2.




**Входные данные**

Во входном файле INPUT.TXT записано 2 символа – координаты клетки, где стоит конь.

**Выходные данные**

В выходной файл OUTPUT.TXT в произвольном порядке выведите все координаты клеток, на которые за один ход может попасть конь, находящийся на заданной клетке.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | e2 | c1<br>c3<br>d4<br>f4<br>g1<br>g3 |
| 2 | a1 | b3<br>c2 |

## 417. Даты

*(Время: 1 сек. Память: 16 Мб Сложность: 34%)*

При обработке различной информации часто приходится оперировать с данными о датах. В этой задаче вам нужно составить программу, которая сможет вывести число, месяц и день недели, которые наступят через K дней, начиная с первого января 2008 года. Напомним, что это был вторник.

**Входные данные**

Входной файл INPUT.TXT содержит одно целое число K ($0 \le K \le 1000$) - количество дней, после первого сентября, через которое наступит (или наступила) интересующая дата.

**Выходные данные**

В выходной файл OUTPUT.TXT ваша программа должна записать, какие день недели, число и месяц наступят по прошествии заданного времени. Результаты проверяются автоматически, поэтому вам следует придерживаться формата, показанного в примерах.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 0 | Tuesday, 01.01 |
| 2 | 5 | Sunday, 06.01 |

*Примечание*

По-английски дни недели называются так: Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday.

## 418. Редактор

*(Время: 1 сек. Память: 16 Мб Сложность: 39%)*

Немногие знают, что первые версии текстового редактора из предыдущей задачи были написаны в России программистом Колей. Для этого он несколько месяцев почти не выходил из подвала, где стояли только диван и компьютер.

Вот одна из проблем, с которыми столкнулся Коля в те времена. Во время работы пользователь набирает какой-то текст, а так же может его редактировать. При этом, даже если итоговый результат полностью помещается на экран, в процессе работы отдельные строки могут иметь слишком большую длину. Мы не будем просить вас повторить Колин подвиг и заново написать редактор. Определите, какой максимальной длины строка получалась в течение набора текста, если вам известно, какие клавиши и в каком порядке нажимал пользователь.

**Входные данные**

Во входном файле INPUT.TXT записана строка из различных символов – последовательность кнопок на клавиатуре, которые нажимал пользователь. Переводы строк заменены на символ «\». Первые версии редактора поддерживали три управляющие команды, которые закодированы следующим образом:

«<» – удаление предыдущего символа (если курсор находится в начале строки, и эта строка не первая, то удаляется предшествующий перевод строки);

«^» – перемещение в конец предыдущей строки (игнорируется, если курсор находится на первой строке);

«|» – перемещение в конец следующей строки (игнорируется, если курсор находится на последней строке).

Все остальные символы, содержащиеся в файле, имеют коды от 32 (пробел) и выше и должны пониматься как есть. Число нажатий клавиш не превосходит $10^5$.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите одно целое число – максимальную длину строки, которая была достигнута в течение работы редактора.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | `Hello, World???<<<!` | `15` |
| 2 | `Hello, World?\This is^<!!!| a sample.` | `17` |

## 419. Палиндром

*(Время: 1 сек. Память: 16 Мб Сложность: 41%)*

Одно из известных развлечений со словами – составление палиндромов. Палиндромом называется предложение, которое, после удаления из него всех пробелов и знаков препинания, читается одинаково справа налево и слева направо. Создатели одного известного текстового редактора пишут новую версию модуля для проверки орфографии. Они хотят реализовать возможность вывода подсказки для пользователя на тот случай, если он допустил опечатку при наборе какого-нибудь палиндрома. Конечно же, они решили обратиться именно к вам.

Более точно, по заданной строке нужно определить, может ли она быть результатом замены, удаления или добавления одного символа в некотором палиндроме. При этом строчные и прописные латинские буквы не различаются, а все остальные символы должны игнорироваться.

**Входные данные**

Во входном файле INPUT.TXT содержится заданная строка. Гарантируется, что она содержит хотя бы одну букву. Длина строки не превосходит $10^5$.

**Выходные данные**

В первой строке выходного файла OUTPUT.TXT выведите YES, если строка может быть получена каким-нибудь из описанных выше преобразований из некоторого палиндрома, и NO в противном случае. В случае положительного ответа во второй строке выведите какой-нибудь из палиндромов, в которых мог допустить опечатку пользователь.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | Never odd or even | YES<br>NeVErODDoREVen |
| 2 | Eat it! | NO |
| 3 | Mums are not set as a test on Erasmus. | YES<br>Sums are not set as a test on Erasmus. |

## 420. Химическая формула

*(Время: 1 сек. Память: 16 Мб Сложность: 36%)*

Запись химической реакции всегда содержит описания нескольких веществ. В свою очередь, описание одного химического вещества – строка, в которой входящие в него атомы химических элементов перечисляются в определенном порядке. При этом последовательности из двух и более одинаковых атомов, идущих подряд, группируются: записывается сокращенное название химического элемента и количество одинаковых элементов подряд. Например, вместо HH пишут H2. Обозначения химических элементов состоят из одной или двух латинских букв, из которых первая – прописная, а вторая – строчная. В этой задаче не будут рассматриваться более сложные правила. Например, не используются скобки. Вы должны проверить, что заданная последовательность символов подходит под данное выше описание формулы химического вещества. При этом не нужно рассматривать корректность заданной строки, исходя из каких-либо других соображений, даже если они продиктованы здравым смыслом.

**Входные данные**

В единственной строке входного файла INPUT.TXT записана последовательность символов, содержащая только цифры и строчные и прописные латинские буквы. Гарантируется, что в последовательности перед каждой строчной буквой идет прописная, а все однобуквенные и двухбуквенные подстроки, начинающиеся с прописной буквы – правильные обозначения химических элементов (поэтому здесь даже не приводится их список). Длина последовательности не превосходит 1000 символов.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите одно слово YES, если данная строка подходит под упрощенное описание формулы химического вещества из условия и NO, если не подходит.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | OHNaOHNa | YES |
| 2 | H2O | YES |
| 3 | HH | NO |
| 4 | CHC | YES |

## 421. Треугольники - 2

*(Время: 1 сек. Память: 16 Мб Сложность: 37%)*

Когда Вова зашел в класс математики, на школьной доске было в беспорядке нарисовано некоторое количество треугольников. Вове очень понравилась идея разрисовывания доски треугольниками, и он принялся изучать ситуацию более внимательно. Вова подумал, что наиболее простой и быстрый способ ровно нарисовать такое количество треугольников - вырезать из какого-нибудь материала шаблон и обводить его на доске мелом. Кроме того, Вова предположил, что тот, кто рисовал треугольники, должно быть, очень спешил, и потому перемещал шаблон, не отрывая никакую его часть от доски.

Помогите Вове проверить его гипотезу, если он готов сообщить вам тщательно измеренные координаты вершин треугольников.

**Входные данные**

В первой строке входного файла INPUT.TXT записано одно целое число – количество треугольников N ($2 \le N \le 100$). Последующие N строк описывают треугольники, строка i+1 содержит 3 пары положительных целых чисел, не превосходящих 10000 – координаты вершин треугольника номер i.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите ответ на задачу: YES, если все треугольники можно было нарисовать, обводя один шаблон и не отрывая этот шаблон от доски, и NO в противном случае.

**Примеры**

| № | INPUT.TXT | OUTPUT.TXT |
|---|---|---|
| 1 | 3<br>1 1 1 2 2 1<br>2 1 2 2 1 2<br>1 1 2 1 2 2 | YES |
| 2 | 2<br>1 1 4 1 2 2<br>1 1 4 1 3 2 | NO |

## Рекомендации учащимся

При выборе задач рекомендуем пользоваться следующей таблицей. В ней для каждой задачи приведена тематика и сложность. Первый параметр позволяет выбирать задачи из тех разделов, которые, как вы считаете, вами недостаточно усвоены. Второй параметр позволяет оценить ваши силы, к тому же по этому параметру рассчитывается ваш рейтинг. Для вашего удобства в первой колонке отмечайте свои успехи в решении задач.

Считаем, что вы можете считать себя подготовленным для победы на муниципальной олимпиаде, если решили примерно 100 задач с общим рейтингом 3000 баллов. Также, считаем, что для достойного выступления на окружной олимпиаде эти показатели должны быть в два раза выше.

| + | № | Задача | Тема | Сложность |
|---|---|---|---|---|
| | 1 | A+B | Задачи для начинающих | 2% |
| | 2 | Сумма | Целочисленная арифметика | 19% |
| | 3 | Пятью пять - двадцать пять! | Целочисленная арифметика | 8% |
| | 4 | Игра | Задачи для начинающих | 4% |
| | 5 | Статистика | Сортировка и последовательности | 25% |
| | 6 | Шахматы | Целочисленная арифметика | 18% |
| | 7 | Золото племени АББА | Длинная арифметика | 40% |
| | 8 | Арифметика | Задачи для начинающих | 5% |
| | 9 | Домашнее задание | Сортировка и последовательности | 27% |
| | 10 | Уравнение | Простая математика | 17% |
| | 11 | Зайчик | Динамическое программирование | 68% |
| | 12 | Дачники | Геометрия | 45% |
| | 13 | Быки и коровы | Разбор строк | 26% |
| | 14 | НОК | Целочисленная арифметика | 24% |
| | 15 | Дороги | Теория графов | 18% |
| | 16 | Лесенка | Рекурсия, перебор | 55% |
| | 17 | Поле чудес | Сортировка и последовательности | 31% |
| | 18 | Факториал | Длинная арифметика | 42% |
| | 19 | Ферзь, ладья и конь | Простая математика | 34% |
| | 20 | Пилообразная последователь-ность | Сортировка и последовательности | 38% |
| | 21 | Зарплата | Задачи для начинающих | 4% |
| | 22 | Единицы | Целочисленная арифметика | 16% |
| | 23 | Гадание | Целочисленная арифметика | 13% |
| | 24 | Вырубка деревьев | Рекурсия, перебор | 46% |
| | 25 | Больше-меньше | Задачи для начинающих | 3% |
| | 26 | Две окружности | Геометрия | 17% |
| | 27 | Художник | Двумерные массивы | 31% |
| | 28 | Симметрия | Геометрия | 19% |
| | 29 | Компьютерная игра | Динамическое программирование | 38% |
| | 30 | Часы | Математическое моделирование | 35% |
| | 31 | Неподвижные точки | Рекурсия, перебор | 57% |
| | 32 | Годовой баланс | Сортировка и последовательности | 33% |
| | 33 | Два бандита | Задачи для начинающих | 4% |
| | 34 | Секретное сообщение | Сортировка и последовательности | 46% |
| | 35 | Конечные автоматы | Задачи для начинающих | 11% |
| | 36 | Постулат Бертрана | Целочисленная арифметика | 38% |
| | 37 | Сжимающий оператор | Геометрия | 34% |

| + | № | Задача | Тема | Сложность |
|---|---|---|---|---|
| | 38 | Игра - 2 | Динамическое программирование | 60% |
| | 39 | Волосатый бизнес | Математическое моделирование | 32% |
| | 40 | 2^N | Длинная арифметика | 30% |
| | 41 | Цифровая сортировка | Сортировка и последовательности | 29% |
| | 42 | Драконы | Математическое моделирование | 40% |
| | 43 | Нули | Задачи для начинающих | 16% |
| | 44 | Стрелки | Разбор строк | 20% |
| | 45 | Произведение цифр | Целочисленная арифметика | 47% |
| | 46 | Число E | Задачи для начинающих | 10% |
| | 47 | Наилучший делитель | Целочисленная арифметика | 26% |
| | 48 | Наихудший делитель | Целочисленная арифметика | 23% |
| | 49 | Шаблоны | Комбинаторика | 35% |
| | 50 | Строки | Разбор строк | 44% |
| | 51 | Факториалы!!! | Целочисленная арифметика | 26% |
| | 52 | Счастливый билет | Целочисленная арифметика | 12% |
| | 53 | Раскраска таблицы умножения | Целочисленная арифметика | 32% |
| | 54 | Теория игр | Двумерные массивы | 28% |
| | 55 | Фонарики | Геометрия | 31% |
| | 56 | Живой Журнал | Сортировка и последовательности | 29% |
| | 57 | Компьютерная сеть | Математическое моделирование | 33% |
| | 58 | Проверка на симпатичность | Двумерные массивы | 28% |
| | 59 | Несложное вычисление | Целочисленная арифметика | 25% |
| | 60 | Сверхпростые числа | Целочисленная арифметика | 39% |
| | 61 | Баскетбол | Задачи для начинающих | 5% |
| | 62 | Клетки | Простая математика | 15% |
| | 63 | Загадка | Простая математика | 18% |
| | 64 | Простой ряд | Сортировка и последовательности | 27% |
| | 65 | Расстояние Хэмминга | Сортировка и последовательности | 32% |
| | 66 | Клавиатура | Задачи для начинающих | 11% |
| | 67 | Маска подсетей | Целочисленная арифметика | 33% |
| | 68 | Дом - Школа - Дом | Задачи для начинающих | 21% |
| | 69 | N-угольное колесо | Геометрия | 28% |
| | 70 | Степень строки | Разбор строк | 30% |
| | 71 | Две кучки камней | Комбинаторика | 43% |
| | 72 | Анаграмма | Сортировка и последовательности | 45% |
| | 73 | Расшифровка | Сортировка и последовательности | 28% |
| | 74 | Прыжки с шестом | Математическое моделирование | 38% |
| | 75 | Сумма произведений | Длинная арифметика | 65% |
| | 76 | Музей | Сортировка и последовательности | 59% |
| | 77 | Нолики | Комбинаторика | 63% |
| | 78 | Бутылки | Математическое моделирование | 48% |
| | 79 | Последняя цифра A^B | Целочисленная арифметика | 21% |
| | 80 | Тождество | Разбор строк | 26% |
| | 81 | Арбузы | Задачи для начинающих | 14% |
| | 82 | Пересечение множеств | Сортировка и последовательности | 34% |
| | 83 | Симпатичные узоры | Динамическое программирование | 66% |
| | 84 | Выпуклая оболочка | Двумерные массивы | 27% |
| | 85 | Единичный НОД | Целочисленная арифметика | 23% |
| | 86 | Головоломка про ферзей | Простая математика | 15% |
| | 87 | Строки - 2 | Сортировка и последовательности | 32% |

| + | № | Задача | Тема | Сложность |
|---|---|---|---|---|
| | 88 | Судоку | Двумерные массивы | 27% |
| | 89 | Быстрый поезд | Сортировка и последовательности | 25% |
| | 90 | Треугольные страны | Геометрия | 37% |
| | 91 | Две последовательности | Сортировка и последовательности | 29% |
| | 92 | Журавлики | Задачи для начинающих | 7% |
| | 93 | Боги | Сортировка и последовательности | 26% |
| | 94 | Принц и дракон | Математическое моделирование | 22% |
| | 95 | Нумеролог | Разбор строк | 24% |
| | 96 | Винни-пух | Математическое моделирование | 41% |
| | 97 | Заповедники | Теория графов | 60% |
| | 98 | Игра в числа | Математическое моделирование | 28% |
| | 99 | Лабиринт | Теория графов | 57% |
| | 100 | Счастливые билеты | Динамическое программирование | 86% |
| | 101 | Магараджа | Рекурсия, перебор | 70% |
| | 102 | Треугольник | Геометрия | 32% |
| | 103 | Снова A+B | Длинная арифметика | 35% |
| | 104 | Шаблон | Рекурсия, перебор | 65% |
| | 105 | Раскопки | Рекурсия, перебор | 76% |
| | 106 | Монетки | Задачи для начинающих | 8% |
| | 107 | Красивые номера | Рекурсия, перебор | 62% |
| | 108 | Неглухой телефон | Задачи для начинающих | 1% |
| | 109 | A / B | Длинная арифметика | 43% |
| | 110 | Красивые числа | Простая математика | 78% |
| | 111 | Игра «Пуговицы» | Математическое моделирование | 48% |
| | 112 | Армия | Сортировка и последовательности | 58% |
| | 113 | Фермер | Динамическое программирование | 46% |
| | 114 | Без двух нулей подряд | Динамическое программирование | 37% |
| | 115 | Прямоугольник | Динамическое программирование | 42% |
| | 116 | Фермер - 2 | Динамическое программирование | 60% |
| | 117 | Опасная зона | Геометрия | 57% |
| | 118 | Задача Иосифа Флавия | Математическое моделирование | 19% |
| | 119 | Сортировка времени | Сортировка и последовательности | 13% |
| | 120 | Минимальный путь в таблице | Динамическое программирование | 32% |
| | 121 | Гвоздики | Динамическое программирование | 34% |
| | 122 | Максимальная подпоследова-тельность | Динамическое программирование | 38% |
| | 123 | Восстановление скобок | Динамическое программирование | 54% |
| | 124 | Светофорчики | Теория графов | 25% |
| | 125 | Цветной дождь | Теория графов | 26% |
| | 126 | Издевательство | Теория графов | 28% |
| | 127 | Путь | Теория графов | 40% |
| | 128 | Один конь | Теория графов | 43% |
| | 129 | Табличка | Теория графов | 40% |
| | 130 | Два коня | Теория графов | 55% |
| | 131 | Перепись | Задачи для начинающих | 15% |
| | 132 | Алгоритм Дейкстры | Теория графов | 47% |
| | 133 | Заправки | Теория графов | 49% |
| | 134 | Автобусы | Теория графов | 50% |
| | 135 | Алгоритм Флойда | Теория графов | 36% |
| | 136 | Алгоритм Флойда - 2 | Теория графов | 39% |

| + | № | Задача | Тема | Сложность |
|---|---|---|---|---|
| | 137 | Существование пути | Теория графов | 65% |
| | 138 | Алгоритм Форда-Беллмана | Теория графов | 38% |
| | 139 | Лабиринт знаний | Теория графов | 45% |
| | 140 | Цикл отрицательного веса | Теория графов | 46% |
| | 141 | Дерево | Теория графов | 42% |
| | 142 | Минимальный каркас | Теория графов | 53% |
| | 143 | A-B | Длинная арифметика | 44% |
| | 144 | A*B | Длинная арифметика | 37% |
| | 145 | A div B | Длинная арифметика | 40% |
| | 146 | Длинный корень | Длинная арифметика | 67% |
| | 147 | Числа Фибоначчи | Целочисленная арифметика | 16% |
| | 148 | НОД | Целочисленная арифметика | 15% |
| | 149 | Разворот | Задачи для начинающих | 9% |
| | 150 | Друзья | Теория графов | 41% |
| | 151 | Банкет | Теория графов | 54% |
| | 152 | Построение | Теория графов | 62% |
| | 153 | Монетки - 2 | Рекурсия, перебор | 51% |
| | 154 | Сумма кубов | Рекурсия, перебор | 52% |
| | 155 | Резисторы | Рекурсия, перебор | 59% |
| | 156 | Шахматы - 2 | Комбинаторика | 46% |
| | 157 | Карточки | Комбинаторика | 48% |
| | 158 | Великий комбинатор | Комбинаторика | 60% |
| | 159 | Обратная перестановка | Комбинаторика | 25% |
| | 160 | Степень перестановки | Комбинаторика | 46% |
| | 161 | Восстановление перестановки | Комбинаторика | 44% |
| | 162 | Манхэттэнский полицейский | Простая математика | 56% |
| | 163 | Уравнение для 5 класса! | Разбор строк | 25% |
| | 164 | Счастливый билет - 2 | Разбор строк | 26% |
| | 165 | Только вправо или вниз | Динамическое программирование | 32% |
| | 166 | Сообщество роботов | Математическое моделирование | 30% |
| | 167 | Количество треугольников | Комбинаторика | 51% |
| | 168 | Натуральный ряд чисел | Разбор строк | 28% |
| | 169 | Магазин | Теория графов | 34% |
| | 170 | Разложение числа | Целочисленная арифметика | 35% |
| | 171 | Количество делителей | Рекурсия, перебор | 50% |
| | 172 | Деление с остатком | Длинная арифметика | 39% |
| | 173 | Число - палиндром | Целочисленная арифметика | 29% |
| | 174 | Свадьба | Сортировка и последовательности | 32% |
| | 175 | Наручные часы | Математическое моделирование | 37% |
| | 176 | Скобочки | Динамическое программирование | 69% |
| | 177 | Склад | Математическое моделирование | 58% |
| | 178 | Преобразование последователь-ности | Сортировка и последовательности | 42% |
| | 179 | Последовательность | Динамическое программирование | 47% |
| | 180 | Счастливая страница | Целочисленная арифметика | 46% |
| | 181 | Космический мусорщик | Динамическое программирование | 52% |
| | 182 | Прямоугольник - 2 | Геометрия | 27% |
| | 183 | Энты | Динамическое программирование | 39% |
| | 184 | Рабочее время | Сортировка и последовательности | 26% |
| | 185 | Скачки | Теория графов | 32% |

| + | № | Задача | Тема | Сложность |
|---|---|---|---|---|
| | 186 | Субботник | Динамическое программирование | 44% |
| | 187 | Пчелка | Динамическое программирование | 53% |
| | 188 | День рождения | Комбинаторика | 75% |
| | 189 | Перестановка по номеру | Комбинаторика | 47% |
| | 190 | По размещению! | Комбинаторика | 56% |
| | 191 | Гладкие числа | Комбинаторика | 60% |
| | 192 | Следующая перестановка ... | Комбинаторика | 46% |
| | 193 | Поиск прямоугольников | Структуры данных | 34% |
| | 194 | Фотограф-зануда | Динамическое программирование | 55% |
| | 195 | Эния | Задачи для начинающих | 3% |
| | 196 | Спираль | Двумерные массивы | 38% |
| | 197 | Змейка | Двумерные массивы | 40% |
| | 198 | Система линейных уравнений | Двумерные массивы | 57% |
| | 199 | Римские числа | Разбор строк | 50% |
| | 200 | Марсианские факториалы | Комбинаторика | 77% |
| | 201 | Пакетная обработка процессов | Математическое моделирование | 48% |
| | 202 | Поиск подстроки | Разбор строк | 44% |
| | 203 | Сдвиг текста | Разбор строк | 51% |
| | 204 | Циклическая строка | Разбор строк | 53% |
| | 205 | Таймер | Простая математика | 31% |
| | 206 | Домой на электричках | Теория графов | 40% |
| | 207 | Клад | Простая математика | 28% |
| | 208 | Забавная игра | Целочисленная арифметика | 30% |
| | 209 | Целые точки | Геометрия | 64% |
| | 210 | Степень | Целочисленная арифметика | 59% |
| | 211 | Игра с фишками | Теория графов | 52% |
| | 212 | Деревни | Теория графов | 79% |
| | 213 | Подсчет баллов | Математическое моделирование | 42% |
| | 214 | Великая сеча | Динамическое программирование | 58% |
| | 215 | Водостоки | Теория графов | 55% |
| | 216 | Коллекционирование этикеток | Комбинаторика | 61% |
| | 217 | Еловая аллея | Динамическое программирование | 47% |
| | 218 | Шашки | Математическое моделирование | 45% |
| | 219 | Симпатичные таблицы | Теория графов | 75% |
| | 220 | Мышка с колесиком | Простая математика | 54% |
| | 221 | Левый лабиринт | Теория графов | 49% |
| | 222 | Дремучий лес | Геометрия | 50% |
| | 223 | Анаграммер | Динамическое программирование | 65% |
| | 224 | Наибольшее произведение | Сортировка и последовательности | 36% |
| | 225 | Покупка билетов | Динамическое программирование | 43% |
| | 226 | Перегоны | Математическое моделирование | 50% |
| | 227 | Сломанный калькулятор | Динамическое программирование | 62% |
| | 228 | Валютные махинации | Динамическое программирование | 39% |
| | 229 | Двухтуровая олимпиада | Математическое моделирование | 46% |
| | 230 | Луч света в темном царстве | Теория графов | 63% |
| | 231 | Распаковка строки | Разбор строк | 25% |
| | 232 | Дремучий лес - 2 | Геометрия | 77% |
| | 233 | Автобусная экскурсия | Задачи для начинающих | 14% |
| | 234 | Сапер | Двумерные массивы | 28% |
| | 235 | Робот К-79 | Математическое моделирование | 30% |

| + | № | Задача | Тема | Сложность |
|---|---|---|---|---|
| | 236 | Многочлен | Разбор строк | 41% |
| | 237 | Головоломка | Жадный алгоритм | 35% |
| | 238 | Побег с космической станции | Теория графов | 52% |
| | 239 | Узор | Динамическое программирование | 80% |
| | 240 | Кубическая гостиница | Жадный алгоритм | 60% |
| | 241 | Праздники | Математическое моделирование | 39% |
| | 242 | Раскраска плиток | Математическое моделирование | 66% |
| | 243 | Маскарад | Динамическое программирование | 63% |
| | 244 | Билетики | Сортировка и последовательности | 37% |
| | 245 | Сплоченная команда | Сортировка и последовательности | 42% |
| | 246 | Вагоны | Сортировка и последовательности | 28% |
| | 247 | Кафе | Математическое моделирование | 46% |
| | 248 | EuroEnglish | Разбор строк | 55% |
| | 249 | Скобки | Разбор строк | 61% |
| | 250 | Двоякие числа | Целочисленная арифметика | 27% |
| | 251 | Калах | Математическое моделирование | 57% |
| | 252 | Сортировка масс | Сортировка и последовательности | 36% |
| | 253 | Часы с боем | Математическое моделирование | 25% |
| | 254 | Выборы жрецов | Структуры данных | 28% |
| | 255 | Представление чисел | Целочисленная арифметика | 40% |
| | 256 | Гексагон | Математическое моделирование | 38% |
| | 257 | Кубическое уравнение | Простая математика | 56% |
| | 258 | Скорая помощь | Математическое моделирование | 42% |
| | 259 | A-функция от строчки | Динамическое программирование | 59% |
| | 260 | Олимпиада по алхимии | Теория графов | 53% |
| | 261 | Лотерея | Математическое моделирование | 71% |
| | 262 | Коммерческий калькулятор | Структуры данных | 52% |
| | 263 | Метро | Задачи для начинающих | 16% |
| | 264 | Оттепель | Задачи для начинающих | 17% |
| | 265 | Шахматная доска | Двумерные массивы | 36% |
| | 266 | Кассы | Сортировка и последовательности | 39% |
| | 267 | Ксерокопии | Простая математика | 38% |
| | 268 | Почти палиндром | Динамическое программирование | 55% |
| | 269 | Тормозной механизм | Жадный алгоритм | 40% |
| | 270 | Java vs C++ | Разбор строк | 44% |
| | 271 | Число Фибоначчи | Целочисленная арифметика | 20% |
| | 272 | Сумма максимума и минимума | Задачи для начинающих | 26% |
| | 273 | Вычеркивание | Рекурсия, перебор | 28% |
| | 274 | Дружные числа | Целочисленная арифметика | 25% |
| | 275 | Делимость на 7 | Целочисленная арифметика | 42% |
| | 276 | Разбиение на части | Простая математика | 21% |
| | 277 | Школьная алгебра | Задачи для начинающих | 27% |
| | 278 | Вычислительная биология | Разбор строк | 28% |
| | 279 | Скобочки - 2 | Разбор строк | 57% |
| | 280 | Количество делителей - 2 | Рекурсия, перебор | 51% |
| | 281 | Игра с монеткой | Комбинаторика | 39% |
| | 282 | Прямоугольники | Динамическое программирование | 47% |
| | 283 | Рунные слова | Разбор строк | 25% |
| | 284 | Подмассив массива | Задачи для начинающих | 15% |
| | 285 | Костер | Математическое моделирование | 34% |

| + | № | Задача | Тема | Сложность |
|---|---|---|---|---|
| | 286 | Больше-меньше - 2 | Длинная арифметика | 30% |
| | 287 | Профессор | Разбор строк | 36% |
| | 288 | Комментарии | Разбор строк | 52% |
| | 289 | Делители | Целочисленная арифметика | 76% |
| | 290 | База террористов | Двумерные массивы | 42% |
| | 291 | Словарь | Разбор строк | 31% |
| | 292 | Простой цифровой корень | Целочисленная арифметика | 36% |
| | 293 | Налоги | Задачи для начинающих | 20% |
| | 294 | Болты и гайки | Задачи для начинающих | 17% |
| | 295 | Шифровка | Разбор строк | 29% |
| | 296 | Лиса Алиса и кот Базилио | Целочисленная арифметика | 22% |
| | 297 | Кругляши | Задачи для начинающих | 16% |
| | 298 | Стрелок | Геометрия | 28% |
| | 299 | Волейбол | Комбинаторика | 56% |
| | 300 | Радар | Математическое моделирование | 38% |
| | 301 | Код | Математическое моделирование | 46% |
| | 302 | Города | Теория графов | 65% |
| | 303 | Цифры | Математическое моделирование | 27% |
| | 304 | Волейбол - 2 | Комбинаторика | 68% |
| | 305 | Морской бой | Динамическое программирование | 60% |
| | 306 | Танец | Математическое моделирование | 45% |
| | 307 | Атлеты | Теория графов | 50% |
| | 308 | Вода | Динамическое программирование | 70% |
| | 309 | К-удивительные числа | Целочисленная арифметика | 30% |
| | 310 | Рамка из клеток | Простая математика | 33% |
| | 311 | Сумма факториалов | Длинная арифметика | 45% |
| | 312 | Арифметическая прогрессия | Простая математика | 15% |
| | 313 | Ежеминутные автобусы | Сортировка и последовательности | 30% |
| | 314 | Лексикографический порядок чисел | Сортировка и последовательности | 31% |
| | 315 | Наименьшая система счисления | Задачи для начинающих | 26% |
| | 316 | Телеграфный перевод | Математическое моделирование | 29% |
| | 317 | Подарки Деда Мороза | Рекурсия, перебор | 27% |
| | 318 | Следующее число | Целочисленная арифметика | 36% |
| | 319 | Точки отрезка | Геометрия | 42% |
| | 320 | Коридор | Динамическое программирование | 38% |
| | 321 | Разные цифры | Целочисленная арифметика | 32% |
| | 322 | Слово | Разбор строк | 28% |
| | 323 | Гипотеза Гольбаха | Целочисленная арифметика | 30% |
| | 324 | Четырехзначный палиндром | Задачи для начинающих | 10% |
| | 325 | Мы с конем вдвоем по полю пойдем | Рекурсия, перебор | 31% |
| | 326 | Преобразование последователь-ности - 2 | Сортировка и последовательности | 29% |
| | 327 | В одном шаге от счастья | Целочисленная арифметика | 16% |
| | 328 | Точки на костях | Комбинаторика | 25% |
| | 329 | Лесенка-2 | Динамическое программирование | 37% |
| | 330 | Телепортация | Простая математика | 30% |
| | 331 | Время прибытия | Задачи для начинающих | 26% |
| | 332 | Минимальная стоимость проезда | Динамическое программирование | 40% |

| + | № | Задача | Тема | Сложность |
|---|---|---|---|---|
| | 333 | Общие цифры | Целочисленная арифметика | 28% |
| | 334 | Китайские часы | Динамическое программирование | 37% |
| | 335 | Трипростые числа | Динамическое программирование | 40% |
| | 336 | Лифт | Задачи для начинающих | 20% |
| | 337 | Лампочки | Сортировка и последовательности | 94% |
| | 338 | Лоскутки | Теория графов | 50% |
| | 339 | Мероприятие | Простая математика | 31% |
| | 340 | Коробки | Простая математика | 19% |
| | 341 | Числовая последовательность | Сортировка и последовательности | 35% |
| | 342 | Вписанная окружность | Геометрия | 68% |
| | 343 | Укладка плиток | Двумерные массивы | 35% |
| | 344 | Ближайшие точки | Сортировка и последовательности | 38% |
| | 345 | Рекурсия | Теория графов | 46% |
| | 346 | Сумма двух чисел | Рекурсия, перебор | 46% |
| | 347 | Покер | Сортировка и последовательности | 33% |
| | 348 | Пересечение отрезков | Геометрия | 49% |
| | 349 | Простые числа | Целочисленная арифметика | 28% |
| | 350 | Перестановки | Рекурсия, перебор | 44% |
| | 351 | Прыжки по буквам | Динамическое программирование | 56% |
| | 352 | Дробь | Рекурсия, перебор | 39% |
| | 353 | Треугольники | Геометрия | 41% |
| | 354 | Разложение на простые множители | Целочисленная арифметика | 27% |
| | 355 | Перестановки - 2 | Рекурсия, перебор | 46% |
| | 356 | Копилка | Динамическое программирование | 49% |
| | 357 | Делимость на 11 | Целочисленная арифметика | 22% |
| | 358 | Забор в парке | Целочисленная арифметика | 42% |
| | 359 | Змейка - 2 | Математическое моделирование | 38% |
| | 360 | Максимальная тройка | Двумерные массивы | 33% |
| | 361 | Подстроки из одинаковых букв | Разбор строк | 52% |
| | 362 | Открытка и конверт | Геометрия | 50% |
| | 363 | Длинное произведение | Длинная арифметика | 46% |
| | 364 | Совершенные числа | Целочисленная арифметика | 51% |
| | 365 | Разложение на слагаемые | Рекурсия, перебор | 54% |
| | 366 | Выражение | Рекурсия, перебор | 56% |
| | 367 | Степень - 2 | Длинная арифметика | 45% |
| | 368 | Маршрут | Динамическое программирование | 38% |
| | 369 | Гангстеры | Динамическое программирование | 44% |
| | 370 | Площадь многоугольника | Геометрия | 48% |
| | 371 | Дружественные числа | Целочисленная арифметика | 58% |
| | 372 | Скобки - 2 | Рекурсия, перебор | 52% |
| | 373 | Маршрут - 2 | Динамическое программирование | 47% |
| | 374 | Выпуклая оболочка - 2 | Геометрия | 55% |
| | 375 | Системы счисления | Целочисленная арифметика | 53% |
| | 376 | День рождения - 2 | Математическое моделирование | 32% |
| | 377 | Закраска прямой | Структуры данных | 48% |
| | 378 | Суммы | Динамическое программирование | 62% |
| | 379 | Игра с датой | Динамическое программирование | 54% |
| | 380 | Площадь прямоугольников | Геометрия | 50% |
| | 381 | Lines | Теория графов | 44% |

| + | № | Задача | Тема | Сложность |
|---|---|---|---|---|
| | 382 | Покраска лабиринта | Рекурсия, перебор | 46% |
| | 383 | Красивые числа - 2 | Целочисленная арифметика | 26% |
| | 384 | Числа Фибоначчи - 3 | Целочисленная арифметика | 52% |
| | 385 | Развлечения с измерителем | Жадный алгоритм | 33% |
| | 386 | Генерация тестов | Математическое моделирование | 37% |
| | 387 | Левая рекурсия | Задачи для начинающих | 20% |
| | 388 | Седловые точки | Двумерные массивы | 28% |
| | 389 | К коду Грея | Математическое моделирование | 50% |
| | 390 | Треугольная область | Геометрия | 41% |
| | 391 | Взлом хеш-функции | Жадный алгоритм | 35% |
| | 392 | Сдвиг перестановки | Сортировка и последовательности | 24% |
| | 393 | Плейлист | Сортировка и последовательности | 48% |
| | 394 | Апельсины | Целочисленная арифметика | 31% |
| | 395 | Произведение цифр - 2 | Целочисленная арифметика | 28% |
| | 396 | Точки и отрезки | Сортировка и последовательности | 62% |
| | 397 | Качество строки | Разбор строк | 52% |
| | 398 | Сумма - 2 | Жадный алгоритм | 25% |
| | 399 | Жук | Математическое моделирование | 30% |
| | 400 | Коробка | Рекурсия, перебор | 50% |
| | 401 | Шары и коробки | Динамическое программирование | 52% |
| | 402 | Коттеджный поселок | Геометрия | 75% |
| | 403 | Обмен валюты | Комбинаторика | 70% |
| | 404 | Игра с камнями | Математическое моделирование | 63% |
| | 405 | Туристическое агентство | Теория графов | 80% |
| | 406 | Криптография | Сортировка и последовательности | 38% |
| | 407 | Сдача | Динамическое программирование | 44% |
| | 408 | Письмо | Разбор строк | 31% |
| | 409 | Железная дорога | Задачи для начинающих | 26% |
| | 410 | Цифровое колдовство | Целочисленная арифметика | 75% |
| | 411 | Квадратное уравнение | Простая математика | 28% |
| | 412 | Ферзь и король | Геометрия | 32% |
| | 413 | Военная база | Двумерные массивы | 42% |
| | 414 | Расследование | Теория графов | 40% |
| | 415 | Подпись | Жадный алгоритм | 30% |
| | 416 | Шахматный конь | Задачи для начинающих | 25% |
| | 417 | Даты | Математическое моделирование | 34% |
| | 418 | Редактор | Структуры данных | 39% |
| | 419 | Палиндром | Разбор строк | 41% |
| | 420 | Химическая формула | Разбор строк | 36% |
| | 421 | Треугольники - 2 | Геометрия | 37% |

## Рекомендации учителям

При выборе задач рекомендуем пользоваться следующей таблицей. В ней задачи упорядочены по тематике, а потом по сложности. Тематика задач идет в порядке усложнения – от тем, достаточных для выступления учащегося на школьной и муниципальной олимпиаде, до сложных тем, задачи по которым используются на окружной олимпиаде и олимпиадах более высокого уровня. Для вашего удобства в таблице имеется первая колонка, в которой рекомендуем отмечать прорешенные вами или вашими учениками задачи.

| + | № | Задача | Тема | Сложность |
|---|---|---|---|---|
| | 108 | Неглухой телефон | Задачи для начинающих | 1% |
| | 1 | A+B | Задачи для начинающих | 2% |
| | 25 | Больше-меньше | Задачи для начинающих | 3% |
| | 195 | Эния | Задачи для начинающих | 3% |
| | 4 | Игра | Задачи для начинающих | 4% |
| | 21 | Зарплата | Задачи для начинающих | 4% |
| | 33 | Два бандита | Задачи для начинающих | 4% |
| | 8 | Арифметика | Задачи для начинающих | 5% |
| | 61 | Баскетбол | Задачи для начинающих | 5% |
| | 92 | Журавлики | Задачи для начинающих | 7% |
| | 106 | Монетки | Задачи для начинающих | 8% |
| | 149 | Разворот | Задачи для начинающих | 9% |
| | 46 | Число Е | Задачи для начинающих | 10% |
| | 324 | Четырехзначный палиндром | Задачи для начинающих | 10% |
| | 35 | Конечные автоматы | Задачи для начинающих | 11% |
| | 66 | Клавиатура | Задачи для начинающих | 11% |
| | 81 | Арбузы | Задачи для начинающих | 14% |
| | 233 | Автобусная экскурсия | Задачи для начинающих | 14% |
| | 131 | Перепись | Задачи для начинающих | 15% |
| | 284 | Подмассив массива | Задачи для начинающих | 15% |
| | 43 | Нули | Задачи для начинающих | 16% |
| | 263 | Метро | Задачи для начинающих | 16% |
| | 297 | Кругляши | Задачи для начинающих | 16% |
| | 264 | Оттепель | Задачи для начинающих | 17% |
| | 294 | Болты и гайки | Задачи для начинающих | 17% |
| | 293 | Налоги | Задачи для начинающих | 20% |
| | 336 | Лифт | Задачи для начинающих | 20% |
| | 387 | Левая рекурсия | Задачи для начинающих | 20% |
| | 68 | Дом - Школа - Дом | Задачи для начинающих | 21% |
| | 416 | Шахматный конь | Задачи для начинающих | 25% |
| | 272 | Сумма максимума и минимума | Задачи для начинающих | 26% |
| | 315 | Наименьшая система счисления | Задачи для начинающих | 26% |
| | 331 | Время прибытия | Задачи для начинающих | 26% |
| | 409 | Железная дорога | Задачи для начинающих | 26% |
| | 277 | Школьная алгебра | Задачи для начинающих | 27% |
| | 3 | Пятью пять - двадцать пять! | Целочисленная арифметика | 8% |
| | 52 | Счастливый билет | Целочисленная арифметика | 12% |
| | 23 | Гадание | Целочисленная арифметика | 13% |
| | 148 | НОД | Целочисленная арифметика | 15% |
| | 22 | Единицы | Целочисленная арифметика | 16% |

| + | № | Задача | Тема | Сложность |
|---|---|---|---|---|
| | 147 | Числа Фибоначчи | Целочисленная арифметика | 16% |
| | 327 | В одном шаге от счастья | Целочисленная арифметика | 16% |
| | 6 | Шахматы | Целочисленная арифметика | 18% |
| | 2 | Сумма | Целочисленная арифметика | 19% |
| | 271 | Число Фибоначчи | Целочисленная арифметика | 20% |
| | 79 | Последняя цифра A^B | Целочисленная арифметика | 21% |
| | 296 | Лиса Алиса и кот Базилио | Целочисленная арифметика | 22% |
| | 357 | Делимость на 11 | Целочисленная арифметика | 22% |
| | 48 | Наихудший делитель | Целочисленная арифметика | 23% |
| | 85 | Единичный НОД | Целочисленная арифметика | 23% |
| | 14 | НОК | Целочисленная арифметика | 24% |
| | 59 | Несложное вычисление | Целочисленная арифметика | 25% |
| | 274 | Дружные числа | Целочисленная арифметика | 25% |
| | 47 | Наилучший делитель | Целочисленная арифметика | 26% |
| | 51 | Факториалы!!! | Целочисленная арифметика | 26% |
| | 383 | Красивые числа - 2 | Целочисленная арифметика | 26% |
| | 250 | Двоякие числа | Целочисленная арифметика | 27% |
| | 354 | Разложение на простые множители | Целочисленная арифметика | 27% |
| | 333 | Общие цифры | Целочисленная арифметика | 28% |
| | 349 | Простые числа | Целочисленная арифметика | 28% |
| | 395 | Произведение цифр - 2 | Целочисленная арифметика | 28% |
| | 173 | Число - палиндром | Целочисленная арифметика | 29% |
| | 208 | Забавная игра | Целочисленная арифметика | 30% |
| | 309 | К-удивительные числа | Целочисленная арифметика | 30% |
| | 323 | Гипотеза Гольбаха | Целочисленная арифметика | 30% |
| | 394 | Апельсины | Целочисленная арифметика | 31% |
| | 53 | Раскраска таблицы умножения | Целочисленная арифметика | 32% |
| | 321 | Разные цифры | Целочисленная арифметика | 32% |
| | 67 | Маска подсетей | Целочисленная арифметика | 33% |
| | 170 | Разложение числа | Целочисленная арифметика | 35% |
| | 292 | Простой цифровой корень | Целочисленная арифметика | 36% |
| | 318 | Следующее число | Целочисленная арифметика | 36% |
| | 36 | Постулат Бертрана | Целочисленная арифметика | 38% |
| | 60 | Сверхпростые числа | Целочисленная арифметика | 39% |
| | 255 | Представление чисел | Целочисленная арифметика | 40% |
| | 275 | Делимость на 7 | Целочисленная арифметика | 42% |
| | 358 | Забор в парке | Целочисленная арифметика | 42% |
| | 180 | Счастливая страница | Целочисленная арифметика | 46% |
| | 45 | Произведение цифр | Целочисленная арифметика | 47% |
| | 364 | Совершенные числа | Целочисленная арифметика | 51% |
| | 384 | Числа Фибоначчи - 3 | Целочисленная арифметика | 52% |
| | 375 | Системы счисления | Целочисленная арифметика | 53% |
| | 371 | Дружественные числа | Целочисленная арифметика | 58% |
| | 210 | Степень | Целочисленная арифметика | 59% |
| | 410 | Цифровое колдовство | Целочисленная арифметика | 75% |
| | 289 | Делители | Целочисленная арифметика | 76% |
| | 62 | Клетки | Простая математика | 15% |
| | 86 | Головоломка про ферзей | Простая математика | 15% |
| | 312 | Арифметическая прогрессия | Простая математика | 15% |

| + | № | Задача | Тема | Сложность |
|---|---|---|---|---|
| | 10 | Уравнение | Простая математика | 17% |
| | 63 | Загадка | Простая математика | 18% |
| | 340 | Коробки | Простая математика | 19% |
| | 276 | Разбиение на части | Простая математика | 21% |
| | 207 | Клад | Простая математика | 28% |
| | 411 | Квадратное уравнение | Простая математика | 28% |
| | 330 | Телепортация | Простая математика | 30% |
| | 205 | Таймер | Простая математика | 31% |
| | 339 | Мероприятие | Простая математика | 31% |
| | 310 | Рамка из клеток | Простая математика | 33% |
| | 19 | Ферзь, ладья и конь | Простая математика | 34% |
| | 267 | Ксерокопии | Простая математика | 38% |
| | 220 | Мышка с колесиком | Простая математика | 54% |
| | 162 | Манхэттэнский полицейский | Простая математика | 56% |
| | 257 | Кубическое уравнение | Простая математика | 56% |
| | 110 | Красивые числа | Простая математика | 78% |
| | 26 | Две окружности | Геометрия | 17% |
| | 28 | Симметрия | Геометрия | 19% |
| | 182 | Прямоугольник - 2 | Геометрия | 27% |
| | 69 | N-угольное колесо | Геометрия | 28% |
| | 298 | Стрелок | Геометрия | 28% |
| | 55 | Фонарики | Геометрия | 31% |
| | 102 | Треугольник | Геометрия | 32% |
| | 412 | Ферзь и король | Геометрия | 32% |
| | 37 | Сжимающий оператор | Геометрия | 34% |
| | 90 | Треугольные страны | Геометрия | 37% |
| | 421 | Треугольники - 2 | Геометрия | 37% |
| | 353 | Треугольники | Геометрия | 41% |
| | 390 | Треугольная область | Геометрия | 41% |
| | 319 | Точки отрезка | Геометрия | 42% |
| | 12 | Дачники | Геометрия | 45% |
| | 370 | Площадь многоугольника | Геометрия | 48% |
| | 348 | Пересечение отрезков | Геометрия | 49% |
| | 222 | Дремучий лес | Геометрия | 50% |
| | 362 | Открытка и конверт | Геометрия | 50% |
| | 380 | Площадь прямоугольников | Геометрия | 50% |
| | 374 | Выпуклая оболочка - 2 | Геометрия | 55% |
| | 117 | Опасная зона | Геометрия | 57% |
| | 209 | Целые точки | Геометрия | 64% |
| | 342 | Вписанная окружность | Геометрия | 68% |
| | 402 | Коттеджный поселок | Геометрия | 75% |
| | 232 | Дремучий лес - 2 | Геометрия | 77% |
| | 84 | Выпуклая оболочка | Двумерные массивы | 27% |
| | 88 | Судоку | Двумерные массивы | 27% |
| | 54 | Теория игр | Двумерные массивы | 28% |
| | 58 | Проверка на симпатичность | Двумерные массивы | 28% |
| | 234 | Сапер | Двумерные массивы | 28% |
| | 388 | Седловые точки | Двумерные массивы | 28% |
| | 27 | Художник | Двумерные массивы | 31% |
| | 360 | Максимальная тройка | Двумерные массивы | 33% |

| + | № | Задача | Тема | Сложность |
|---|---|---|---|---|
| | 343 | Укладка плиток | Двумерные массивы | 35% |
| | 265 | Шахматная доска | Двумерные массивы | 36% |
| | 196 | Спираль | Двумерные массивы | 38% |
| | 197 | Змейка | Двумерные массивы | 40% |
| | 290 | База террористов | Двумерные массивы | 42% |
| | 413 | Военная база | Двумерные массивы | 42% |
| | 198 | Система линейных уравнений | Двумерные массивы | 57% |
| | 120 | Минимальный путь в таблице | Динамическое программирование | 32% |
| | 165 | Только вправо или вниз | Динамическое программирование | 32% |
| | 121 | Гвоздики | Динамическое программирование | 34% |
| | 114 | Без двух нулей подряд | Динамическое программирование | 37% |
| | 329 | Лесенка-2 | Динамическое программирование | 37% |
| | 334 | Китайские часы | Динамическое программирование | 37% |
| | 29 | Компьютерная игра | Динамическое программирование | 38% |
| | 122 | Максимальная подпоследовательность | Динамическое программирование | 38% |
| | 320 | Коридор | Динамическое программирование | 38% |
| | 368 | Маршрут | Динамическое программирование | 38% |
| | 183 | Энты | Динамическое программирование | 39% |
| | 228 | Валютные махинации | Динамическое программирование | 39% |
| | 332 | Минимальная стоимость проезда | Динамическое программирование | 40% |
| | 335 | Трипростые числа | Динамическое программирование | 40% |
| | 115 | Прямоугольник | Динамическое программирование | 42% |
| | 225 | Покупка билетов | Динамическое программирование | 43% |
| | 186 | Субботник | Динамическое программирование | 44% |
| | 369 | Гангстеры | Динамическое программирование | 44% |
| | 407 | Сдача | Динамическое программирование | 44% |
| | 113 | Фермер | Динамическое программирование | 46% |
| | 179 | Последовательность | Динамическое программирование | 47% |
| | 217 | Еловая аллея | Динамическое программирование | 47% |
| | 282 | Прямоугольники | Динамическое программирование | 47% |
| | 373 | Маршрут - 2 | Динамическое программирование | 47% |
| | 356 | Копилка | Динамическое программирование | 49% |
| | 181 | Космический мусорщик | Динамическое программирование | 52% |
| | 401 | Шары и коробки | Динамическое программирование | 52% |
| | 187 | Пчелка | Динамическое программирование | 53% |
| | 123 | Восстановление скобок | Динамическое программирование | 54% |
| | 379 | Игра с датой | Динамическое программирование | 54% |
| | 194 | Фотограф-зануда | Динамическое программирование | 55% |
| | 268 | Почти палиндром | Динамическое программирование | 55% |
| | 351 | Прыжки по буквам | Динамическое программирование | 56% |
| | 214 | Великая сеча | Динамическое программирование | 58% |
| | 259 | А-функция от строчки | Динамическое программирование | 59% |
| | 38 | Игра - 2 | Динамическое программирование | 60% |
| | 116 | Фермер - 2 | Динамическое программирование | 60% |
| | 305 | Морской бой | Динамическое программирование | 60% |
| | 227 | Сломанный калькулятор | Динамическое программирование | 62% |
| | 378 | Суммы | Динамическое программирование | 62% |
| | 243 | Маскарад | Динамическое программирование | 63% |

| + | № | Задача | Тема | Сложность |
|---|---|---|---|---|
| | 223 | Анаграммер | Динамическое программирование | 65% |
| | 83 | Симпатичные узоры | Динамическое программирование | 66% |
| | 11 | Зайчик | Динамическое программирование | 68% |
| | 176 | Скобочки | Динамическое программирование | 69% |
| | 308 | Вода | Динамическое программирование | 70% |
| | 239 | Узор | Динамическое программирование | 80% |
| | 100 | Счастливые билеты | Динамическое программирование | 86% |
| | 40 | 2^N | Длинная арифметика | 30% |
| | 286 | Больше-меньше - 2 | Длинная арифметика | 30% |
| | 103 | Снова A+B | Длинная арифметика | 35% |
| | 144 | A*B | Длинная арифметика | 37% |
| | 172 | Деление с остатком | Длинная арифметика | 39% |
| | 7 | Золото племени АББА | Длинная арифметика | 40% |
| | 145 | A div B | Длинная арифметика | 40% |
| | 18 | Факториал | Длинная арифметика | 42% |
| | 109 | A / B | Длинная арифметика | 43% |
| | 143 | A-B | Длинная арифметика | 44% |
| | 311 | Сумма факториалов | Длинная арифметика | 45% |
| | 367 | Степень - 2 | Длинная арифметика | 45% |
| | 363 | Длинное произведение | Длинная арифметика | 46% |
| | 75 | Сумма произведений | Длинная арифметика | 65% |
| | 146 | Длинный корень | Длинная арифметика | 67% |
| | 398 | Сумма - 2 | Жадный алгоритм | 25% |
| | 415 | Подпись | Жадный алгоритм | 30% |
| | 385 | Развлечения с измерителем | Жадный алгоритм | 33% |
| | 237 | Головоломка | Жадный алгоритм | 35% |
| | 391 | Взлом хеш-функции | Жадный алгоритм | 35% |
| | 269 | Тормозной механизм | Жадный алгоритм | 40% |
| | 240 | Кубическая гостиница | Жадный алгоритм | 60% |
| | 159 | Обратная перестановка | Комбинаторика | 25% |
| | 328 | Точки на костях | Комбинаторика | 25% |
| | 49 | Шаблоны | Комбинаторика | 35% |
| | 281 | Игра с монеткой | Комбинаторика | 39% |
| | 71 | Две кучки камней | Комбинаторика | 43% |
| | 161 | Восстановление перестановки | Комбинаторика | 44% |
| | 156 | Шахматы - 2 | Комбинаторика | 46% |
| | 160 | Степень перестановки | Комбинаторика | 46% |
| | 192 | Следующая перестановка ... | Комбинаторика | 46% |
| | 189 | Перестановка по номеру | Комбинаторика | 47% |
| | 157 | Карточки | Комбинаторика | 48% |
| | 167 | Количество треугольников | Комбинаторика | 51% |
| | 190 | По размещению! | Комбинаторика | 56% |
| | 299 | Волейбол | Комбинаторика | 56% |
| | 158 | Великий комбинатор | Комбинаторика | 60% |
| | 191 | Гладкие числа | Комбинаторика | 60% |
| | 216 | Коллекционирование этикеток | Комбинаторика | 61% |
| | 77 | Нолики | Комбинаторика | 63% |
| | 304 | Волейбол - 2 | Комбинаторика | 68% |
| | 403 | Обмен валюты | Комбинаторика | 70% |
| | 188 | День рождения | Комбинаторика | 75% |

| + | № | Задача | Тема | Сложность |
|---|---|---|---|---|
| | 200 | Марсианские факториалы | Комбинаторика | 77% |
| | 118 | Задача Иосифа Флавия | Математическое моделирование | 19% |
| | 94 | Принц и дракон | Математическое моделирование | 22% |
| | 253 | Часы с боем | Математическое моделирование | 25% |
| | 303 | Цифры | Математическое моделирование | 27% |
| | 98 | Игра в числа | Математическое моделирование | 28% |
| | 316 | Телеграфный перевод | Математическое моделирование | 29% |
| | 166 | Сообщество роботов | Математическое моделирование | 30% |
| | 235 | Робот К-79 | Математическое моделирование | 30% |
| | 399 | Жук | Математическое моделирование | 30% |
| | 39 | Волосатый бизнес | Математическое моделирование | 32% |
| | 376 | День рождения - 2 | Математическое моделирование | 32% |
| | 57 | Компьютерная сеть | Математическое моделирование | 33% |
| | 285 | Костер | Математическое моделирование | 34% |
| | 417 | Даты | Математическое моделирование | 34% |
| | 30 | Часы | Математическое моделирование | 35% |
| | 175 | Наручные часы | Математическое моделирование | 37% |
| | 386 | Генерация тестов | Математическое моделирование | 37% |
| | 74 | Прыжки с шестом | Математическое моделирование | 38% |
| | 256 | Гексагон | Математическое моделирование | 38% |
| | 300 | Радар | Математическое моделирование | 38% |
| | 359 | Змейка - 2 | Математическое моделирование | 38% |
| | 241 | Праздники | Математическое моделирование | 39% |
| | 42 | Драконы | Математическое моделирование | 40% |
| | 96 | Винни-пух | Математическое моделирование | 41% |
| | 213 | Подсчет баллов | Математическое моделирование | 42% |
| | 258 | Скорая помощь | Математическое моделирование | 42% |
| | 218 | Шашки | Математическое моделирование | 45% |
| | 306 | Танец | Математическое моделирование | 45% |
| | 229 | Двухтуровая олимпиада | Математическое моделирование | 46% |
| | 247 | Кафе | Математическое моделирование | 46% |
| | 301 | Код | Математическое моделирование | 46% |
| | 78 | Бутылки | Математическое моделирование | 48% |
| | 111 | Игра «Пуговицы» | Математическое моделирование | 48% |
| | 201 | Пакетная обработка процессов | Математическое моделирование | 48% |
| | 226 | Перегоны | Математическое моделирование | 50% |
| | 389 | К коду Грея | Математическое моделирование | 50% |
| | 251 | Калах | Математическое моделирование | 57% |
| | 177 | Склад | Математическое моделирование | 58% |
| | 404 | Игра с камнями | Математическое моделирование | 63% |
| | 242 | Раскраска плиток | Математическое моделирование | 66% |
| | 261 | Лотерея | Математическое моделирование | 71% |
| | 44 | Стрелки | Разбор строк | 20% |
| | 95 | Нумеролог | Разбор строк | 24% |
| | 163 | Уравнение для 5 класса! | Разбор строк | 25% |
| | 231 | Распаковка строки | Разбор строк | 25% |
| | 283 | Рунные слова | Разбор строк | 25% |
| | 13 | Быки и коровы | Разбор строк | 26% |
| | 80 | Тождество | Разбор строк | 26% |
| | 164 | Счастливый билет - 2 | Разбор строк | 26% |

| + | № | Задача | Тема | Сложность |
|---|---|---|---|---|
| | 168 | Натуральный ряд чисел | Разбор строк | 28% |
| | 278 | Вычислительная биология | Разбор строк | 28% |
| | 322 | Слово | Разбор строк | 28% |
| | 295 | Шифровка | Разбор строк | 29% |
| | 70 | Степень строки | Разбор строк | 30% |
| | 291 | Словарь | Разбор строк | 31% |
| | 408 | Письмо | Разбор строк | 31% |
| | 287 | Профессор | Разбор строк | 36% |
| | 420 | Химическая формула | Разбор строк | 36% |
| | 236 | Многочлен | Разбор строк | 41% |
| | 419 | Палиндром | Разбор строк | 41% |
| | 50 | Строки | Разбор строк | 44% |
| | 202 | Поиск подстроки | Разбор строк | 44% |
| | 270 | Java vs C++ | Разбор строк | 44% |
| | 199 | Римские числа | Разбор строк | 50% |
| | 203 | Сдвиг текста | Разбор строк | 51% |
| | 288 | Комментарии | Разбор строк | 52% |
| | 361 | Подстроки из одинаковых букв | Разбор строк | 52% |
| | 397 | Качество строки | Разбор строк | 52% |
| | 204 | Циклическая строка | Разбор строк | 53% |
| | 248 | EuroEnglish | Разбор строк | 55% |
| | 279 | Скобочки - 2 | Разбор строк | 57% |
| | 249 | Скобки | Разбор строк | 61% |
| | 317 | Подарки Деда Мороза | Рекурсия, перебор | 27% |
| | 273 | Вычеркивание | Рекурсия, перебор | 28% |
| | 325 | Мы с конем вдвоем по полю пойдем | Рекурсия, перебор | 31% |
| | 352 | Дробь | Рекурсия, перебор | 39% |
| | 350 | Перестановки | Рекурсия, перебор | 44% |
| | 24 | Вырубка деревьев | Рекурсия, перебор | 46% |
| | 346 | Сумма двух чисел | Рекурсия, перебор | 46% |
| | 355 | Перестановки - 2 | Рекурсия, перебор | 46% |
| | 382 | Покраска лабиринта | Рекурсия, перебор | 46% |
| | 171 | Количество делителей | Рекурсия, перебор | 50% |
| | 400 | Коробка | Рекурсия, перебор | 50% |
| | 153 | Монетки - 2 | Рекурсия, перебор | 51% |
| | 280 | Количество делителей - 2 | Рекурсия, перебор | 51% |
| | 154 | Сумма кубов | Рекурсия, перебор | 52% |
| | 372 | Скобки - 2 | Рекурсия, перебор | 52% |
| | 365 | Разложение на слагаемые | Рекурсия, перебор | 54% |
| | 16 | Лесенка | Рекурсия, перебор | 55% |
| | 366 | Выражение | Рекурсия, перебор | 56% |
| | 31 | Неподвижные точки | Рекурсия, перебор | 57% |
| | 155 | Резисторы | Рекурсия, перебор | 59% |
| | 107 | Красивые номера | Рекурсия, перебор | 62% |
| | 104 | Шаблон | Рекурсия, перебор | 65% |
| | 101 | Магараджа | Рекурсия, перебор | 70% |
| | 105 | Раскопки | Рекурсия, перебор | 76% |
| | 119 | Сортировка времени | Сортировка и последовательности | 13% |

| + | № | Задача | Тема | Сложность |
|---|---|---|---|---|
| | 392 | Сдвиг перестановки | Сортировка и последовательности | 24% |
| | 5 | Статистика | Сортировка и последовательности | 25% |
| | 89 | Быстрый поезд | Сортировка и последовательности | 25% |
| | 93 | Боги | Сортировка и последовательности | 26% |
| | 184 | Рабочее время | Сортировка и последовательности | 26% |
| | 9 | Домашнее задание | Сортировка и последовательности | 27% |
| | 64 | Простой ряд | Сортировка и последовательности | 27% |
| | 73 | Расшифровка | Сортировка и последовательности | 28% |
| | 246 | Вагоны | Сортировка и последовательности | 28% |
| | 41 | Цифровая сортировка | Сортировка и последовательности | 29% |
| | 56 | Живой Журнал | Сортировка и последовательности | 29% |
| | 91 | Две последовательности | Сортировка и последовательности | 29% |
| | 326 | Преобразование последовательности - 2 | Сортировка и последовательности | 29% |
| | 313 | Ежеминутные автобусы | Сортировка и последовательности | 30% |
| | 17 | Поле чудес | Сортировка и последовательности | 31% |
| | 314 | Лексикографический порядок чисел | Сортировка и последовательности | 31% |
| | 65 | Расстояние Хэмминга | Сортировка и последовательности | 32% |
| | 87 | Строки - 2 | Сортировка и последовательности | 32% |
| | 174 | Свадьба | Сортировка и последовательности | 32% |
| | 32 | Годовой баланс | Сортировка и последовательности | 33% |
| | 347 | Покер | Сортировка и последовательности | 33% |
| | 82 | Пересечение множеств | Сортировка и последовательности | 34% |
| | 341 | Числовая последовательность | Сортировка и последовательности | 35% |
| | 224 | Наибольшее произведение | Сортировка и последовательности | 36% |
| | 252 | Сортировка масс | Сортировка и последовательности | 36% |
| | 244 | Билетики | Сортировка и последовательности | 37% |
| | 20 | Пилообразная последовательность | Сортировка и последовательности | 38% |
| | 344 | Ближайшие точки | Сортировка и последовательности | 38% |
| | 406 | Криптография | Сортировка и последовательности | 38% |
| | 266 | Кассы | Сортировка и последовательности | 39% |
| | 178 | Преобразование последовательности | Сортировка и последовательности | 42% |
| | 245 | Сплоченная команда | Сортировка и последовательности | 42% |
| | 72 | Анаграмма | Сортировка и последовательности | 45% |
| | 34 | Секретное сообщение | Сортировка и последовательности | 46% |
| | 393 | Плейлист | Сортировка и последовательности | 48% |
| | 112 | Армия | Сортировка и последовательности | 58% |
| | 76 | Музей | Сортировка и последовательности | 59% |
| | 396 | Точки и отрезки | Сортировка и последовательности | 62% |
| | 337 | Лампочки | Сортировка и последовательности | 94% |
| | 254 | Выборы жрецов | Структуры данных | 28% |
| | 193 | Поиск прямоугольников | Структуры данных | 34% |
| | 418 | Редактор | Структуры данных | 39% |
| | 377 | Закраска прямой | Структуры данных | 48% |
| | 262 | Коммерческий калькулятор | Структуры данных | 52% |
| | 15 | Дороги | Теория графов | 18% |
| | 124 | Светофорчики | Теория графов | 25% |

| + | № | Задача | Тема | Сложность |
|---|---|---|---|---|
| | 125 | Цветной дождь | Теория графов | 26% |
| | 126 | Издевательство | Теория графов | 28% |
| | 185 | Скачки | Теория графов | 32% |
| | 169 | Магазин | Теория графов | 34% |
| | 135 | Алгоритм Флойда | Теория графов | 36% |
| | 138 | Алгоритм Форда-Беллмана | Теория графов | 38% |
| | 136 | Алгоритм Флойда - 2 | Теория графов | 39% |
| | 127 | Путь | Теория графов | 40% |
| | 129 | Табличка | Теория графов | 40% |
| | 206 | Домой на электричках | Теория графов | 40% |
| | 414 | Расследование | Теория графов | 40% |
| | 150 | Друзья | Теория графов | 41% |
| | 141 | Дерево | Теория графов | 42% |
| | 128 | Один конь | Теория графов | 43% |
| | 381 | Lines | Теория графов | 44% |
| | 139 | Лабиринт знаний | Теория графов | 45% |
| | 140 | Цикл отрицательного веса | Теория графов | 46% |
| | 345 | Рекурсия | Теория графов | 46% |
| | 132 | Алгоритм Дейкстры | Теория графов | 47% |
| | 133 | Заправки | Теория графов | 49% |
| | 221 | Левый лабиринт | Теория графов | 49% |
| | 134 | Автобусы | Теория графов | 50% |
| | 307 | Атлеты | Теория графов | 50% |
| | 338 | Лоскутки | Теория графов | 50% |
| | 211 | Игра с фишками | Теория графов | 52% |
| | 238 | Побег с космической станции | Теория графов | 52% |
| | 142 | Минимальный каркас | Теория графов | 53% |
| | 260 | Олимпиада по алхимии | Теория графов | 53% |
| | 151 | Банкет | Теория графов | 54% |
| | 130 | Два коня | Теория графов | 55% |
| | 215 | Водостоки | Теория графов | 55% |
| | 99 | Лабиринт | Теория графов | 57% |
| | 97 | Заповедники | Теория графов | 60% |
| | 152 | Построение | Теория графов | 62% |
| | 230 | Луч света в темном царстве | Теория графов | 63% |
| | 137 | Существование пути | Теория графов | 65% |
| | 302 | Города | Теория графов | 65% |
| | 219 | Симпатичные таблицы | Теория графов | 75% |
| | 212 | Деревни | Теория графов | 79% |
| | 405 | Туристическое агентство | Теория графов | 80% |

# Проведение олимпиад

На сайте регулярно проводятся олимпиады для школьников. В этом разделе приведено положение об олимпиадах и задачи с разбором их решений олимпиад 2007-2008 учебного года для школьников Ханты-Мансийского автономного округа – Югры.

## 1. Цели

1.1. Смоделировать для школьников и учителей информатики школ систему проведения чемпионатов по олимпиадному программированию.
1.2. Повысить профессиональные навыки участников проекта в области программирования.

## 2. Задачи

2.1. Создать общедоступный портал для проведения олимпиад.
2.2. Привлечь к участию в проекте школьников и учителей.
2.3. Формировать и развивать навыки участников в области программирования.

## 3. Участники

3.1. Участниками проекта могут быть любые пользователи сайта, прошедшие регистрацию в разделе «Регистрация».

## 4. Возможности системы

4.1. Проект дает возможность всем зарегистрированным пользователям сайта участвовать в проводимых соревнованиях, выполнять выставленные задачи, просматривать историю выполнения заданий.
4.2. Для участия в каком-либо соревновании пользователи должны подать заявку до начала соревнования. Перед участием в олимпиадах рекомендуется ознакомиться с системой проверки решений и решить несколько задач с сайта.
4.3. Каждая олимпиада представляет собой набор задач по олимпиадному программированию, которые участники должны выполнить. Время проведения олимпиады ограничено. Все олимпиады разделяются на личные и командные. Для проверки правильности программ используется автоматическая проверка решений.
4.4. Личные олимпиады подразумевают индивидуальное участие, при этом для каждой задачи устанавливается ее значимость, характеризуемая максимальным числом баллов, которые участник может набрать. Здесь учитывается неполное решение, за которое участник получает число баллов, пропорциональное проценту правильно пройденных тестов. Сообщение «Accepted» появляется в случае успешного прохождения 1-го теста. Во время проведения олимпиады в рейтинге, как правило, отображаются только результаты факта принятия решений, а по завершению олимпиады в рейтинге можно видеть окончательные результаты в баллах. Более высокую позицию занимает участник, набравший наибольшее количество баллов в сумме за все задачи, в случае равенства этого значений побеждает тот, кто раньше отправил последнее принятое системой решение.
4.5. В командных олимпиадах засчитываются только верные ответы. В том случае, когда участник отправляет верный ответ не с первой попытки, ошибочные ответы переводятся в штрафные баллы: 20 баллов за каждый неверный ответ. Так же, в штрафные баллы суммируется время, затраченное на решение каждой сданной задачи. Неверные решения при подсчете штрафных баллов не учитываются. В процессе командных соревнований формируется «Рейтинг», где все посетители сайта могут видеть текущие результаты: решенные задачи, время, затраченное на решение задач, а так же штрафные очки. Более высокую позицию в рейтинге занимает тот участник, который решил больше задач. В случае равного количества решенных задач победитель определяется по наименьшему числу штрафных очков.
4.6. В последние минуты олимпиады рейтинг может быть заморожен для сохранения интриги участников.

## 5. Формат задач и результат выполнения

5.1. Задачи создаются, выставляются и изменяются администратором, назначенным супервизором на данную олимпиаду.
5.2. Задачи выполняются только зарегистрированными участниками, отправившими заявку на участие в текущем чемпионате до его начала, а так же имеется возможность отправки решений за любого из участников для администратора олимпиады.
5.3. Каждая олимпиада содержит описание, сроки проведения и набор задач для выполнения. Каждая задача соответствует стандартному формату олимпиадной задачи, описанном в разделе «Новичкам». Количество задач обычно варьируется в пределах от 2 до 12, а время проведения от часа до 5 часов.

## 6. АРМ супервизора

6.1. Супервизор может выставлять, редактировать и удалять олимпиады.
6.2. Супервизор может назначать на каждую олимпиаду администратора и сам является администратором.
6.3. Администратор может осуществлять набор задач для олимпиады, замораживать и размораживать рейтинг, контролировать процесс участия с возможностью просмотра исходного кода всех участников. Так же возможна установка пароля для подачи заявки на участие в олимпиаде и удаление заявок.

# Олимпиады 2007-2008 уч. года для школьников Ханты-Мансийского автономного округа – Югры

Всероссийская олимпиада школьников по информатике проводится в пять этапов: школьный, муниципальный, региональный (окружной), окружной федеральный и заключительный. Материалы двух последних этапов Всероссийской олимпиады школьников по информатике 2007-2008 учебного года можно посмотреть на портале Всероссийских олимпиад школьников по адресу http://www.rusolimp.ru. Ниже приведены задачи (тексты задач смотрите в разделе «Архив задач») и их разборы первых трех этапов олимпиады.

# I-й тур школьной олимпиады

## Задача A. 271.

**Разбор**

Пока очередное число Фибоначчи меньше заданного числа ищем следующее число Фибоначчи. Далее сравниваем найденное число Фибоначчи и заданное число.

**Программа**

```
var
  a, b, c, n : longint;
  k : integer;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(n);
  a:=0; b:=1; k:=1;
  while n>b do
  begin
    c:=a+b; a:=b; b:=c; k:=k+1
  end;
  if n=b then begin writeln(1); write(k) end
         else write(0)
end.
```

## Задача B. 272.

**Разбор**

Пока не конец файла читаем из него очередное число и пересчитываем максимум или минимум.

**Программа**

```
var
  a, max, min : integer;
  n : longint;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  max:=-10000; min:=10000; n:=0;
  while not eof do
  begin
    read(a); n:=n+1;
    if odd(n) then if a<min then min:=a else
              else if a>max then max:=a
  end;
  write(max+min)
end.
```

## Задача C. 061.

**Разбор**

Суммируем количество очков по четвертям для каждой из команд. Сравниваем набранные очки за матч.

**Програма**

```
var
  a, b, c, d, i : integer;
begin
  assign(input, 'input.txt'); reset(input);
  assign(output, 'output.txt'); rewrite(output);
  c:=0; d:=0;
  for i:=1 to 4 do
  begin
    read(a,b);
    c:=c+a; d:=d+b
  end;
  if c=d then writeln('DRAW') else
    if c<d then write(2) else write(1)
end.
```

# II-й тур школьной олимпиады

## Задача A. 273.

**Разбор**

Организуем тройной цикл по всем возможным вариантам.

**Программа**

```
var
  s : string;
  a : array [0..999] of byte;
  i, j, k : integer;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(s);
  for j:=0 to 999 do a[j]:=0;
  for i:=1 to length(s)-2 do
    for j:=i+1 to length(s)-1 do
      for k:=j+1 to length(s) do
```

```
        a[((ord(s[i])-48)*10+ord(s[j])-48)*10+ord(s[k])-48]:=1;
  i:=0;
  for j:=100 to 999 do i:=i+a[j];
  write(i)
end.
```

## Задача B. 274.

**Разбор**

Для каждого из чисел в массиве пометим встречающиеся цифры. Далее сравниваем массивы.

**Программа**

```
var
  a, b : longint;
  c, d : array [0..9] of byte;
  i, k : byte;
  t : boolean;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(k);
  while k>0 do
  begin
    k:=k-1;
    read(a,b);
    for i:=0 to 9 do begin c[i]:=0; d[i]:=0 end;
    while a>0 do
    begin
      c[a mod 10]:=1;
      a:=a div 10
    end;
    while b>0 do
    begin
      d[b mod 10]:=1;
      b:=b div 10
    end;
    t:=true;
    for i:=0 to 9 do
      t:=t and (c[i]=d[i]);
    if t then write('YES')
         else write('NO');
    if k>0 then writeln
  end
end.
```

## Задача C. 092.

**Разбор**

Если Петя и Сережа сделали по $x$ журавликов, то Катя сделала $4x$ журавликов. Вместе они сделали $6x$ журавликов.

**Программа**

```
var
  s : longint;
begin
  assign(input, 'input.txt'); reset(input);
  assign(output, 'output.txt'); rewrite(output);
  read(s);
  s:=s div 6;
  write(s,' ',4*s,' ',s)
end.
```

# I-й тур муниципальной олимпиады

## Задача A. 311.

**Разбор**

Так как факториал быстро растущая целочисленная функция, то для получения суммы факториалов необходимо использовать длинную арифметику. Таким образом, требуется реализовать две операции длинной арифметики: сложение и умножение. Для упрощения заметим, что 1!+2!+…+N!=1+2*(1+3*(…(1+N)…)). Это позволяет использовать вместо сложения более просто программируемую операцию добавления единицы.

**Программа**

```
var
  n, p, i, k, l : integer;
  a : array [1..1000] of integer;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(n);
  k:=1; a[k]:=1;
  for i:=n downto 2 do
  begin
    p:=0;
    for l:=1 to k do
    begin p:=p+a[l]*i; a[l]:=p mod 10; p:=p div 10 end;
    while p>0 do begin k:=k+1; a[k]:=p mod 10; p:=p div 10 end;
    l:=1; while a[l]=9 do begin a[l]:=0; l:=l+1 end;
    a[l]:=a[l]+1; if l>k then k:=l
  end;
  for i:=k downto 1 do write(a[i])
end.
```

## Задача B. 309.

**Разбор**

Организуем перебор всех чисел от 1 до заданного. Для каждого числа проверяем его K-удивительность.

**Программа**

```
var
  k, i, j, m, s : longint;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(k);
  s:=0;
  for i:=1 to k do
  begin
    j:=i; m:=0;
    while j>0 do
    begin
      m:=m*10+j mod 10;
      j:=j div 10
    end;
    if i+m=k then s:=s+1
  end;
  write(s)
end.
```

## Задача C. 310.

**Разбор**

Рамку можно покрыть плитками A×1, если A=1 или A=2 или одна из сторон кратна A, а вторая при делении на A дает остаток 2, или каждая при делении на A дает остаток 1.

**Программа**

```
var
  k : integer;
  x, y, a : longint;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(k);
  while k>0 do
  begin
    k:=k-1;
    readln(x,y,a);
    if (a<3) or (x mod a=0) and (y mod a=2) or
       (x mod a=2) and (y mod a=0) or
       (x mod a=1) and (y mod a=1) then write(1) else write(0)
  end
end.
```

# II-й тур муниципальной олимпиады

## Задача A. 317.

**Разбор**

Математически задача сводится к нахождению количества неотрицательных решений ($a$, $b$, $c$) уравнения $a*x+b*y+c*z=w$. Если организовать тройной цикл и проверять выполнение уравнения, то, например, при $x=y=z=1$ и $w=1000$ понадобится $10^9$ проверок. Организуем двойной цикл, например, по переменным $a$ и $b$ и будем проверять разрешимость уравнения относительно $c$.

**Программа**

```
var
  x, y, z, w, a, b, v : integer;
  k : longint;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(x,y,z,w);
  k:=0;
  for a:=0 to w div x do
    for b:=0 to w div y do
    begin
      v:=w-a*x-b*y;
      if (v>=0) and (v mod z=0) then k:=k+1
    end;
  write(k)
end.
```

## Задача B. 318.

**Разбор**

Найдем двоичное разложение заданного числа. Просматривая его слева направо, пропускаем нули, первую единицу заменяем нулем, также далее следующую группу единиц заменяем нулями, подсчитывая количество таких замен, первый встретившийся ноль заменяем на единицу, справа дописываем подсчитанное количество единиц.

**Программа**

```
var
  n : longint;
  i, k, j : integer;
  a : array [1..32] of integer;
begin
  assign(input,'input.txt'); reset(input);
```

```
  assign(output,'output.txt'); rewrite(output);
  read(n); k:=0;
  while n>0 do
  begin k:=k+1; a[k]:=n mod 2; n:=n div 2 end;
  i:=1; while a[i]=0 do i:=i+1;
  a[i]:=0; i:=i+1; j:=0;
  while a[i]=1 do begin a[i]:=0; i:=i+1; j:=j+1 end;
  a[i]:=1; if i>k then k:=i;
  for i:=1 to j do a[i]:=1;
  if k=32 then write('2147483648') else
  begin
    for i:=k downto 1 do n:=n*2+a[i];
    write(n)
  end
end.
```

## Задача C. 319.

**Разбор**

Пусть $(x_1, y_1)$, $(x_2, y_2)$ – координаты концов отрезка. Переместим отрезок и, если нужно, отразим его относительно вертикали и горизонтали так, чтобы его левый нижний конец находился в точке (0, 0), а второй имел координаты $(x, y)=(|x_1-x_2|, |y_1-y_2|)$. Очевидно, что количество «целочисленных» точек на отрезке при этих перемещениях не изменяется. Найдем наибольший общий делитель координат отрезка, d=НОД(x, y). Тогда $x=k\cdot d$, $y=l\cdot d$, то точки (k, l), (2k, 2l), …,$((d-1)\cdot k, (d-1)\cdot l)$ принадлежат отрезку, поэтому всего «целочисленных» точек (d-1)+2=d+1.

**Программа**

```
var
  x1, y1, x2, y2, a, b, c : longint;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(x1,y1,x2,y2);
  a:=abs(x1-x2); b:=abs(y1-y2);
  if a=0 then write(b+1) else
    if b=0 then write(a+1) else
    begin
      while b>0 do
      begin c:=a mod b; a:=b; b:=c end;
      write(a+1)
    end
end.
```

# III-й тур муниципальной олимпиады

## Задача A. 320.

**Разбор**

Задача решается методом динамического программирования. Для этого обозначим через A(n) – количество способов замостить коридор длиной n. Так как для коридора длиной k<m плитки можно укладывать только горизонтально, то A(k)=1. В других случаях плитки в последнем ряду можно положить или горизонтально или вертикально, а тогда A(n)=A(n-1)+A(n-m). Приведенные соотношения позволяют последовательно рассчитать требуемое количество замощений кридора. Ответом будет значение A(N).

**Программа**

```
var
  m, n, i : integer;
  a : array [0..50] of longint;
begin
  assign(input,'input.txt'); reset(input);
```

```
  assign(output,'output.txt'); rewrite(output);
  read(m,n);
  for i:=0 to m-1 do a[i]:=1;
  for i:=m to n do
    a[i]:=a[i-1]+a[i-m];
  write(a[n])
end.
```

## Задача B. 321.

**Разбор**

В цикле от 2 до 36 по основанию системы счисления находим для каждой из цифр этой системы счисления ее количество в записи числа. Далее проверяем требуемое условие: в записи числа нет одинаковых цифр.

**Программа**

```
var
  n, i : longint;
  k, j, m : integer;
  c : array [0..35] of integer;
  t : boolean;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(n);
  for k:=2 to 36 do
  begin
    i:=n; for j:=0 to k-1 do c[j]:=0;
    while i>0 do
    begin
      m:=i mod k; c[m]:=c[m]+1; i:=i div k
    end;
    t:=true;
    for j:=0 to k-1 do t:=t and (c[j]<2);
    if t then write(k,' ')
  end
end.
```

## Задача C. 322.

**Разбор**

Вводим символы из файла, одновременно подсчитывая их номера и числа Фибоначчи. Если номер введенного символа совпадает с очередным числом Фибоначчи, то выводим этот символ.

**Программа**

```
var
  s : char;
  a, b, c, i : integer;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  a:=1; b:=1; i:=0;
  while not eoln do
  begin
    i:=i+1;
    read(s);
    if i=b then
    begin write(s);
      c:=a+b; a:=b; b:=c
    end
  end
end.
```

# Олимпиада Института прикладной математики, информатики и управления ЮГУ «Осень-2007»

Открытые олимпиады Института прикладной математики, информатики и управления (ИПМИУ) Югорского государственного университета традиционно проводятся с 2002 года два раза в год. Для участия в них приглашаются не только студенты, но и школьники. В предыдущие годы в этих олимпиадах участвовали школьники Югорского физико-математического лицея и школ города Ханты-Мансийска. В олимпиаде «Осень-2007» благодаря Интернету смогли принять участие школьники и других образовательных учреждений нашего округа, России и стран ближнего зарубежья. Следующая олимпиада ИПМИУ «Весна-2008» была проведена на сайте http://olymp.irohmao.ru, на котором в дальшейшем эти олимпиады и будут проводиться.

## Задача A. 312.

**Разбор**

Для решения задачи необходимо помнить формулу для $n$-го члена арифметической прогрессии $a_n=a_1+d\cdot(n-1)=a_1+(a_2-a_1)\cdot(n-1)$.

**Программа**

```
var
  a1, a2, n : integer;
  an : longint;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(a1,a2,n);
  an:=a2-a1; an:=a1+an*(n-1);
  write(an)
end.
```

## Задача B. 313.

**Разбор**

Так как маршрутов не больше ста, то в массиве из ста элементов будем хранить информацию о каждом из маршрутов (время прибытия автобуса этого маршрута на остановку). Последовательным просмотром входного файла изменяем значения используемого массива и одновременно находим требуемое время.

**Программа**

```
var
  n, i, m, max : longint;
  t : integer;
  a : array [1..100] of longint;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(n);
  for t:=1 to 100 do a[t]:=0;
  for i:=1 to n do
  begin
    read(t);
    if a[t]=0 then a[t]:=i else
    begin
      m:=i-a[t]; a[t]:=i;
      if m>max then max:=m
    end
  end;
  write(max)
end.
```

## Задача C. 314.

**Разбор**

Числа от 1 до N запишем в массив строк и отсортируем его. Сортировку можно сделать одним из простых методов, например, методом «пузырька». Такой алгоритм не даёт полный результат, так как требует большого количества действий. Конечно, можно применить быструю сортировку, но для уменьшения количества требуемых действий попытаемся найти номер того места, на котором должно располагаться заданное число K. Просмотрев все числа от 1 до N и сравнивая очередное и заданное числа как строки, найдем требуемое.

**Программа**

```
var
  n, i, k, m : longint;
  s, t : string;
begin
  assign(input, 'input.txt'); reset(input);
  assign(output, 'output.txt'); rewrite(output);
  read(n,k); str(k,t); m:=0;
  for i:=1 to n do
  begin
    str(i,s);
    if s<=t then m:=m+1
  end;
  write(m)
end.
```

## Задача D. 315.

**Разбор**

Просмотром строки проверяем наличие в ней только допустимых символов. Одновременно определяем основание допустимой системы счисления. Так как не существует системы счисления с основанием 1, то обрабатываем этот случай дополнительно.

**Программа**

```
var
  s, t : string;
  v : boolean;
  i, m, max : integer;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  readln(s);
  t:='0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ';
  v:=false;
  for i:=1 to length(s) do v:=v or (pos(s[i],t)=0);
  if v then write(-1) else
  begin
    max:=0; for i:=1 to length(s) do
    begin
      m:=pos(s[i],t);
      if m>max then max:=m
    end;
    if max=1 then max:=2;
    write(max)
  end
end.
```

## Задача E. 170.

**Разбор**

Математически задача сводится к решению уравнения в целых числах. Действительно, пусть в разложении $l$ слагаемых, начиная с $a$ – $a$, $a$+1, …, $a$+($l$-1). Тогда $a+a+1+\ldots+a+(l-1)=a\cdot l+l\cdot(l-1)/2$ и имеем следующее уравнение $a\cdot l+l\cdot(l-1)/2=n$. Решая его относительно $a$, получаем $a=(n- l\cdot(l-1)/2)/l$. Так как $a$ должно быть целым, то выбираем только те значения $l$, для которых это так. Из всех решений выбираем то, для которого $l$ максимально.

**Программа**

```
var
  n, l, lmax : longint;
begin
  assign(input, 'input.txt'); reset(input);
  assign(output, 'output.txt'); rewrite(output);
  read(n);
  l:=1; lmax:=1;
  while n-l*(l+1) div 2 >0 do
  begin
    l:=l+1;
    if (n-(l-1)*l div 2) mod l=0 then lmax:=l
  end;
  write(lmax)
end.
```

## Задача F. 316

**Разбор**

Простая задача на целочисленную арифметику.

**Программа**

```
var
  n, x : word;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(n);
  x:=n;
  while x+(x+99)div 100 *7>n do x:=x-1;
  write(x,' ',(x+99)div 100 *7)
end.
```

# Пробный тур окружной олимпиады

Для ознакомления всех участников окружной олимпиады с тестирующей системой был проведен пробный тур. Его результаты не влияли ни на состав участников, ни учитывались при подведении итогов, но позволяли будущим участникам качественно подготовиться к ней.

## Задача A. 327.

**Разбор**

С помощью логической функции определения «счастливости» билета проверяем билеты с меньшим и большим номерами.

**Программа**

```
var
  a : longint;
  k : integer;
function sc(a:longint):boolean;
  var x,y,i: integer;
begin
  x:=0; for i:=1 to 3 do begin x:=x+a mod 10; a:=a div 10 end;
  y:=0; for i:=1 to 3 do begin y:=y+a mod 10; a:=a div 10 end;
  sc:=x=y
end;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(k);
  while k>0 do
  begin
```

```
      k:=k-1;
      read(a);
      if sc(a-1) or sc(a+1) then writeln('Yes') else writeln('No')
  end
end.
```

## Задача B. 329.

**Разбор**

Для решения задачи используем идею динамического программирования. Обозначим через S(i) – наибольшую сумму в оптимальном маршруте Вовы от первой ступеньки до i-й. Имеем S(1)=a(1), S(2)=a(1)+a(2). На i-ю ступеньку он может попасть или с i-1-й или с i-2-й, из этих двух случаев выбираем тот, для которого имеем наибольшее значение, т.е. S(i)=max{S(i-1), S(i-2)}+a(i). Последовательно рассчитаем значения S для i=1, 2, ..,N. Для определения оптимального маршрута просмотрим полученные значения S(i) в обратном порядке.

**Программа**

```
var
  n, i, a, k : integer;
  s : array [0..1000] of longint;
  t : array [1..1000] of integer;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(n); s[0]:=0;
  read(a); s[1]:=a;
  for i:=2 to n do
  begin
       read(a);
       if s[i-2]<s[i-1] then s[i]:=a+s[i-1] else s[i]:=a+s[i-2]
  end;
  writeln(s[n]);
  k:=1; t[k]:=n;
  while n>1 do
  begin
    if s[n-2]<s[n-1] then n:=n-1 else n:=n-2;
    if n>0 then begin k:=k+1; t[k]:=n end
  end;
  for i:=k downto 1 do write(t[i],' ')
end.
```

## Задача C. 330.

**Разбор**

Так как корабль может двигаться только по диагоналям, то за одну секунду он может из одной точки попасть в другую, если эти точки расположены на одной диагонале. Для телепортации через промежуточную точку надо проверить условие целочисленности координат этой точки.

**Программа**

```
var
  x1, y1, x2, y2 : longint;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(x1,y1,x2,y2);
  if (x1+y1=x2+y2)or(x1-y1=x2-y2) then write(1) else
    if ((x2-x1+y1+y2) mod 2=0) then write(2) else write(0)
end.
```

## Задача D. 331.

**Разбор**

Простая задача на ввод и целочисленную арифметику.

**Программа**

```
var
  ho, mo, hp, mp : integer;
  s : string;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  readln(s);
  ho:=(ord(s[1])-48)*10+ord(s[2])-48;
  mo:=(ord(s[4])-48)*10+ord(s[5])-48;
  read(hp,mp);
  hp:=ho+hp; mp:=mo+mp;
  if mp>59 then begin mp:=mp-60; hp:=hp+1 end;
  while hp>23 do hp:=hp-24;
  if hp<10 then write(0); write(hp,':');
  if mp<10 then write(0); write(mp)
end.
```

## Задача E. 332.

**Разбор**

Задача решается методом динамического программирования. Обозначим через S(i) - минимальную стоимость проезда от станции 0 до станции i с возможными пересадками. Сразу заметим, что S(0)=0. До i-й станции можно доехать или электричкой со станции 0 или с пересадками на 1-й, 2-й, …, i-1-й станциях. Из всех этих способов надо выбрать тот, который обеспечивает минимальную стоимость, т.е. имеем S(i)=min{a(0, i), S(1)+a(1,i), S(2)+a(2,i), …, S(i-1)+a(i-1,i)}, где a(k, i) – стоимость проезда от k-й станции до i-й станции. При указанных в задаче ограничениях, для хранения стоимостей проезда от одной до другой станции требуется использовать более компактное представление, чем двумерный массив. В приведенной ниже программе для этого используется одномерный массив.

**Программа**

```
var
  n, i, j, k, l, m, min, n1, n2 : integer;
  a : array [1..31375] of integer;
  s : array [0..250] of integer;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(n);
  n1:=n; n2:=n+1; if n1 mod 2=0 then n1:=n1 div 2 else n2:=n2 div 2;
  for i:=1 to n1*n2 do read(a[i]);
  s[0]:=0;
  for i:=1 to n do
  begin
    min:=32767; k:=i; l:=n-1;
    for j:=0 to i-1 do
    begin
      m:=s[j]+a[k];
      if m<min then min:=m;
      k:=k+1; l:=l-1
    end;
    s[i]:=min
  end;
  write(s[n])
end.
```

## Задача F. 333.

**Разбор**

Для каждого числа найдем множество его цифр. Для каждой цифры определит принадлежность ее всем трем множествам. Для описания множества цифр числа используем массив от 0 до 9, в котором элемент массива равен единице, если цифра есть в числе.

**Программа**

```
var
  a, b, c : array [0..9] of integer;
  s : string;
  i, k : integer;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  readln(s); s:=s+' ';
  for i:=0 to 9 do begin a[i]:=0; b[i]:=0; c[i]:=0 end;
  i:=1; while s[i]<>' ' do begin a[ord(s[i])-48]:=1; i:=i+1 end;
  i:=i+1; while s[i]<>' ' do begin b[ord(s[i])-48]:=1; i:=i+1 end;
  i:=i+1; while s[i]<>' ' do begin c[ord(s[i])-48]:=1; i:=i+1 end;
  k:=0;
  for i:=0 to 9 do
    if a[i]+b[i]+c[i]=3 then begin a[k]:=i; k:=k+1 end;
  write(k); if k>0 then writeln;
  for i:=0 to k-1 do
  begin write(a[i]); if i<k then write(' ') end
end.
```

# I-й тур окружной олимпиады

## Задача A. 344.

**Разбор**

Найдем расстояния от первой точки до второй, третьей, …, n-й, от второй до третьей, …, n-й, …, n-1-й до n-й и из них выберем минимальное расстояние. Такой алгоритм требует n*(n-1)/2 действий и даёт только 40 баллов из 100. Отсортируем массив координат по возрастанию. Тогда требуемое минимальное расстояние можно найти, сравнивая соседние элементы отсортированного массива. Для уменьшения количества действий сортировку массива необходимо сделать улучшенным алгоритмом, например, используя быструю сортировку.

**Программа**

```
var
  n, i, j, c : longint;
  a, b : array [1..100000] of longint;
procedure QuickSort( L, R : longInt );
var i,j,z : longint; x,y : longint;
begin
  i := l; j := r;
  x := a[(l+r) div 2];
  repeat
    while (A[i] < x) do inc(i);
    while (x < A[j]) do dec(j);
    if ( i <= j ) then
    begin
      y:=A[i]; a[i]:=a[j]; a[j]:=y;
      z:=b[i]; b[i]:=b[j]; b[j]:=z;
      inc(i); dec(j);
    end;
  until (i > j);
  if (l < j) then QuickSort(l,j);
  if (i < r) then QuickSort(i,r);
end;
begin
```

```
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  readln(n);
  for i:=1 to n do
  begin read(a[i]); b[i]:=i end;
  QuickSort(1,n);
  c:=maxlongint;
  for i:=1 to n-1 do
    if a[i+1]-a[i]<c then
    begin c:=a[i+1]-a[i]; j:=i end;
  writeln(c);
  write(b[j],' ',b[j+1]);
  close(output)
end.
```

## Задача B. 351.

**Разбор**

Задача на динамическое программирование. Обозначим M[i] – стоимость перехода на i-ю букву. Легко посчитать M[i] для i=1, 2, …, k+1. Если i-я буква совпадаем с первой, то M[i]=0, иначе M[i]=1. Для i=k+2, …, n стоимость M[i] можно посчитать через предыдущие k значений, используя классическую схему динамического программирования. Это требует (n-k)*k действий и при выбранной системе тестов дает только 80 из 100 возможных баллов. Для уменьшения количества действий воспользуемся тем фактом, что в цепочке используются только прописные латинские буквы, которых всего 26 от A до Z. Заведем два массива S и M из 26 элементов. В массиве S будем хранить номер последнего появления соответствующей буквы в цепочке, в массиве M – стоимость перехода на соответстующую букву. Обрабатывая последовательно символы цепочки, будем пересчитывать эти массивы. После обработки последнего символа в массиве M будем иметь необходимый результат. Количество необходимых действий пропорционально длине заданной цепочки n.

**Программа**

```
var
  n, k, i, min : longint;
  c, v, v1 : char;
  S, M : array ['A'..'Z'] of longint;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  readln(n,k);
  read(v1);
  for c:='A' to 'Z' do begin S[c]:=0; M[c]:=maxlongint end;
  for i:=2 to k+1 do
  begin
    read(v);
    if v1=v then M[v]:=0 else M[v]:=1;
    S[v]:=i
  end;
  for i:=k+2 to n do
  begin
    read(v);
    min:=M[v]; v1:=v;
    for c:='A' to 'Z' do
      if M[c]<min then begin min:=M[c]; v1:=c end;
    if v<>v1 then min:=min+1;
    M[v]:=min; S[v]:=i;
    for c:='A' to 'Z' do
      if S[c]+k=i then begin S[c]:=0; M[c]:=maxlongint end;
  end;
  write(M[v])
end.
```

## Задача C. 341.

**Разбор**

Решение данной задачи предполагает последовательное нахождение элементов последовательности, описанной в условии задачи. Заметим, что каждое число последовательности обладает следующим свойством: в десятичной записи все его цифры, кроме первой, равны. Таким образом, для нахождения следующего числа достаточно перебрать первую цифру числа, остальные цифры числа, а также длину числа (количество цифр очередного числа равно или на единицу больше количества цифр предыдущего числа последовательности). После этого необходимо найти наименьшее из чисел, удовлетворяющих требованием задачи (множества цифр очередного числа и предыдущего числа последовательности не должны пересекаться).

**Программа**

```
var
  n, i, a, b, k, c, d : integer;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(n);
  if n<12 then write(n-1) else
  begin
    a:=1; b:=0; k:=1;
    for i:=12 to n do
    begin
      if a<9 then
      begin
        c:=a+1; d:=0; while d in [a, b] do d:=d+1;
      end
      else
      begin
        k:=k+1;
        c:=1; while c in [a,b] do c:=c+1;
        d:=0; while d in [a,b] do d:=d+1
      end;
      a:=c; b:=d
    end;
    write(a); for i:=1 to k do write(b)
  end
end.
```

# II-й тур окружной олимпиады

## Задача D. 352.

**Разбор**

Достаточно простая задача на целочисленную арифметику. Уменьшая k от [n/2] найдем первую несократимую дробь k/(n-k). Для проверки несократимости дроби находим наибольший общий делитель чисел k и n-k. Рекомедуется для этого использовать обобщенный алгоритм Евклида, так как алгоритм Евклида на выбранной системе тестов не дает полный результат.

**Программа**

```
var
  n, k, a, b, c : longint;
  t : boolean;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(n);
  k:=(n+1) div 2; t:=true;
  while t do
  begin
    k:=k-1;
    a:=k; b:=n-k;
```

```
    while b>0 do
    begin c:=a mod b; a:=b; b:=c end;
    t:=a<>1
  end;
  write(k,' ',n-k)
end.
```

## Задача E. 346.

**Разбор**

Основная идея решения данной задачи основана на переборе всех перестановок цифр числа a. Чтобы это сделать, обозначим полученное число $a_{perm}$. Теперь, для того, чтобы найти число $b_{perm}$, которое необходимо получить перестановкой цифр числа b, достаточно вычесть из числа c число $a_{perm}$. Для проверки возможности перестановки цифр числа b таким образом, чтобы получилось число $b_{perm}$, предлагается проверить на равенство мультимножества цифр указанных чисел. Это можно сделать, например, посчитав количество нулей, единиц, двоек … девяток в каждом из сравниваемых чисел.

**Программа**

```
  type Pere=array [byte] of byte;
  var N,i,j : byte;
  X : Pere;
  Yes, Y : boolean;
  a, b, c, ao : longint;
  aa, bb : array [0..9] of byte;
procedure Next(var X:Pere;var Yes:boolean);
  var i:byte;
  procedure Swap(var a,b:byte);
  var c:byte;
  begin c:=a;a:=b;b:=c end;
begin
  i:=N-1;
  while (i>0)and(X[i]>=X[i+1]) do dec(i);
  if i>0 then
  begin
    j:=i+1;
    while (j<N)and(X[j+1]>X[i]) do inc(j);
    Swap(X[i],X[j]);
    for j:=i+1 to (N+i) div 2 do Swap(X[j],X[N-j+i+1]);
    Yes:=true
  end
  else Yes:=false
end;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(a,b,c); n:=0;
  while a>0 do begin n:=n+1; x[n]:=a mod 10; a:=a div 10 end;
  for i:=0 to 9 do bb[i]:=0;
  while b>0 do begin i:=b mod 10; bb[i]:=bb[i]+1; b:=b div 10 end;
  for i:=1 to n-1 do
  for j:=1 to n-i do
  if x[j]>x[j+1] then
  begin a:=x[j]; x[j]:=x[j+1]; x[j+1]:=a end;
  repeat
    a:=0; for i:=1 to n do a:=a*10+x[i];
    ao:=a;
    a:=c-a;
    if a>0 then
    begin
      for i:=0 to 9 do aa[i]:=0;
      while a>0 do begin i:=a mod 10; aa[i]:=aa[i]+1; a:=a div 10 end;
      Y:=true;
```

```
      for i:=0 to 9 do Y:=Y and (aa[i]=bb[i]);
    end;
    Next(X,Yes)
  until not Yes or Y;
  if Y then begin writeln('YES'); write(ao,' ',c-ao) end else write('NO')
end.
```

## Задача F. 353.

**Разбор**

Переборный алгоритм имеет сложность порядка $n^3$, что не позволяет набрать полные баллы. Отсортировав длины отрезков даже простейшим алгоритмом, существенно сокращаем требуемое количество действий.

**Программа**

```
var
  n, i, j, k, a, b, c : integer;
  s : real;
  p, smax : int64;
  w : array [1..1001] of integer;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(n); smax:=0;
  for i:=1 to n do read(w[i]); w[n+1]:=maxint;
  for i:=1 to n-1 do
    for j:=1 to n-i do
    if w[j]>w[j+1] then
    begin c:=w[j]; w[j]:=w[j+1]; w[j+1]:=c end;
  for i:=1 to n-2 do
  begin
    a:=w[i];
    for j:=i+1 to n-1 do
    begin
      b:=w[j];
      k:=j+1; while (k<=n)and(a+b<=w[k]) do k:=k+1;
      while a+b>w[k] do
      begin
        c:=w[k];
        p:=a+b+c;
        p:=p*(p-2*a)*(p-2*b)*(p-2*c);
        if p>smax then smax:=p;
        k:=k+1
      end
    end
  end;
  s:=sqrt(1.0*smax)/4.0;
  if smax=0 then write(0) else write(s:0:3);
  close(output)
end.
```

# Интернет-турнир школ по программированию

Итоговым мероприятием 2007-2008 учебного года в олимпиадной информатике округа стал турнир школ по программированию. В нем приняли участие команды школ, состоящие из учителя и нескольких учеников. Для участия в олимпиаде им на команду был выделен только один логин, с использованием которого команда могла сдавать свои решения. Распределение задач между членами команд полностью возлагалась на учителя и капитана команды. В турнире половина задач (A. 273, E. 320, H. 351, I. 318, J. 311) была взята из проведенных олимпиад с целью проверки умения учителя и школьников работать с задачами прошедших соревнований, то есть умения работать над ошибками. Разборы и программы для этих задач были приведены выше, ищите их по номеру в предыдущих соревнованиях.

## Задача B. 357.

**Разбор**

Из сформулированного в условии задачи признака делимости на 11 следует, что при вводе цифр заданного числа надо суммировать их попеременно со знаками + и -. После этого найденная сумма проверяется на делимость на 11. Для решения можно было использовать схему Горнера.

**Программа**

```
var
  k : integer;
  s : longint;
  c : char;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  k:=1; s:=0;
  while not eoln do
  begin
    read(c);
    s:=s+k*(ord(c)-48);
    k:=-k
  end;
  if s mod 11=0 then write('YES')
                else write('NO');
  close(output)
end.
```

## Задача C. 358.

**Разбор**

Посчитаем количество точек с целочисленными координатами, лежащих на сторонах треугольника. Тогда решением задачи будет сумма этих количеств минус три, так как при таком подсчете вершины треугольника были учтены два раза. Для подсчета количества целочисленных точек на отрезке с концами $(x_1, y_1)$ и $(x_2, y_2)$ преобразуем его в отрезок с концами (0, 0) и $(|x_2-x_1|, |y_2-y_1|)=(a, b)$. Если a=0, то требуемое количество равно b+1. Аналогично, при b=0 имеем a+1. Если же a и b одновременно не равны нулю, то искомое количество равно их наибольшему общему делителю плюс один.

**Программа**

```
var
  x1, y1, x2, y2, x3, y3 : longint;
  d : int64;
function k(x1,y1,x2,y2: longint):longint;
  var a,b,c: longint;
begin
  a:=abs(x2-x1); b:=abs(y2-y1);
  if a=0 then k:=b else
    if b=0 then k:=a else
    begin
      while b>0 do
      begin c:=a mod b; a:=b; b:=c end;
      k:=a
    end
end;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(x1,y1,x2,y2,x3,y3);
  d:=0; d:=d+k(x1,y1,x2,y2)+k(x2,y2,x3,y3)+k(x3,y3,x1,y1);
  write(d);
  close(output)
end.
```

## Задача D. 359.

**Разбор**

Посчитаем, какое число стоит на N-м месте при последовательной их нумерации. После этого уберем каждое десятое число.

**Программа**

```
var
  k, l : int64;
  n, i : longint;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(n);
  if n<3 then write(n) else
  begin
    k:=2; l:=3;
    for i:=2 to n div 2 do
    begin k:=k+2*l; l:=l+2 end;
    if n mod 2=1 then k:=k+l;
    write(10*((k-1) div 9)+(k-1) mod 9+1);
  end;
  close(output)
end.
```

## Задача F. 360.

**Разбор**

Храним три последовательных строки заданной матрицы и пересчитываем тройки при обработке очередной строки.

**Программа**

```
var
  n, i, j, m, max : integer;
  a, b, c : array [1..2000] of integer;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  readln(n);
  for i:=1 to n do read(a[i]);
  max:=a[1]+a[2]+a[3];
  for i:=1 to n-2 do
  begin m:=a[i]+a[i+1]+a[i+2]; if m>max then max:=m end;
  for i:=1 to n do read(b[i]);
  for i:=1 to n-2 do
  begin m:=b[i]+b[i+1]+b[i+2]; if m>max then max:=m end;
  for i:=1 to n-1 do
  begin
    m:=b[i]+b[i+1]+a[i+1]; if m>max then max:=m;
    m:=a[i]+b[i]+b[i+1]; if m>max then max:=m;
    m:=a[i]+b[i+1]+a[i+1]; if m>max then max:=m;
    m:=b[i]+a[i]+a[i+1]; if m>max then max:=m;
  end;
  for j:=3 to n do
  begin
    for i:=1 to n do read(c[i]);
    for i:=1 to n do
    begin m:=a[i]+b[i]+c[i]; if m>max then max:=m end;
    for i:=1 to n-2 do
    begin m:=c[i]+c[i+1]+c[i+2]; if m>max then max:=m end;
    for i:=1 to n-1 do
    begin
      m:=c[i]+c[i+1]+b[i+1]; if m>max then max:=m;
      m:=b[i]+c[i]+c[i+1]; if m>max then max:=m;
      m:=b[i]+c[i+1]+b[i+1]; if m>max then max:=m;
```

```
      m:=c[i]+b[i]+b[i+1]; if m>max then max:=m;
    end;
    a:=b; b:=c
  end;
  write(max);
  close(output)
end.
```

## Задача G. 361.

**Разбор**

Рассмотрим подстроки длины меньшей на 1, 2, , . длины заданной строки, пока не найдем состоящие из одних и тех же букв. Для проверки последнего условия воспользуемся тем фактом, что строка состоит только из малых латинских букв, и в двумерном массиве a накопим количество каждой из букв от начала строки. Это позволяет просто проверять условие наличия у двух подстрок одинаковых букв.

**Программа**

```
var
  a : array ['a'..'z', 0..100] of integer;
  s : string;
  c : char;
  t, v : boolean;
  i, j, l, n : integer;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  readln(s); n:=length(s);
  for c:='a' to 'z' do for j:=0 to n do a[c,j]:=0;
  for i:=1 to n do for j:=i to n do a[s[i],j]:=a[s[i],j]+1;
  t:=true; l:=n;
  while t and (l>1) do
  begin
    l:=l-1;
    for i:=0 to n-l-1 do
      for j:=i+1 to n-l do
      begin
        v:=true;
        for c:='a' to 'z' do
          v:=v and(a[c,i+l]-a[c,i]=a[c,j+l]-a[c,j]);
        if v then t:=false
      end
  end;
  if t then write(0) else write(l);
  close(output)
end.
```

# Тестирующие системы

Аналогичные рассмотренной тестирующие системы существуют на сайтах ведущих вузов России и мира. Их адреса приведены в таблице. Для дальнейшего совершенствования в области олимпиадного программирования советуем порешать задачи на этих сайтах.

| Название | Адрес |
|---|---|
| Портал «Всероссийская олимпиада школьников» | http://www.rusolymp.ru |
| Проект «Дистанционная подготовка» МЦНМО | http://informatics.mccme.ru |
| Олимпиадная информатика, семинар | http://olympiads.ru/sng |
| Центр Интернет-олимпиад, Украина | http://olymp.vinnica.ua |
| Мытищинская школа программистов | http://www.informatics.ru |
| Соревнования по программированию в Самаре | http://contest.samara.ru |
| Дальневосточный университет | http://imcs.dvgu.ru/cats |
| Магнитогорский технический университет | http://www.magtu.ru/OLIMP/tren |
| Московский государственный университет | http://acm.msu.ru |
| Московский физико-технический институт | http://acm.mipt.ru/judge |
| Новосибирский сервер олимпиадных тренировок | http://olimp.iis.nsk.su |
| Саратовский университет | http://acm.sgu.ru |
| Ставропольский государственный университет | http://contest.stavsu.ru |
| Таганрогский технологический институт ЮФУ | http://contester.tsure.ru |
| Томский университет систем управления и радиоэлектроники | http://olimpic.tusur.ru |
| Уральский университет | http://acm.timus.ru |
| Южно-Уральский университет | http://ipc.susu.ac.ru |
| Гданьский Технический университет, Польша | http://www.spoj.pl |
| Дистанционное обучение в Беларуси | http://dll.gsu.unibel.by |
| Кыргызско-Российский Славянский университет, Киргизия | http://www.olymp.krsu.edu.kg |
| Пекинский университет, Китай | http://acm.pku.edu.cn/JudgeOnline/ |
| Университет Вальядолид, Испания | http://acm.uva.es |
| Университет ЖеЙана, Китай | http://acm.zju.edu.cn |
| Университет Тьянжин, Китай | http://acm.tju.edu.cn/toj |
| Хорватские открытые Интернет-олимпиады | http://www.hsin.hr/coci/ |

# Список книг по олимпиадной информатике

Для совершенствования в области олимпиадного программирования авторы рекомендуют использовать приведенную ниже литературу, которую можно найти в Интернете в электронном виде или заказать книги через Интернет-магазины (например, http://www.ozon.ru).


1. Алексеев, А.В. Задачи олимпиад по математике и информатике. Практикум для школьников и студентов. – Ханты-Мансийск: ЮГУ, 2005. – 64 с.
2. Андреева, Е.В. Олимпиады по информатике. Пути к вершине. // Информатика, № 38, 40, 42, 44, 46, 48 за 2001 г., 6, 8, 10, 12, 14, 16 за 2002 г.
3. Андреева, Е.В., Егоров, Ю.Е. Вычислительная геометрия на плоскости. // Информатика, № 39, 40, 43, 44 за 2002 г.
4. Глинка, Н.В. Школьные олимпиады. Информатика. 8-11 классы. – М.: Айрис-пресс, 2007. – 240 с.
5. Горяинов, В.С. и др. Школьные олимпиады: физика, математика, информатика. 8-11 классы. – Ростов н/Д: Феникс, 2004. – 192 с.
6. Долинский, М.С. Алгоритмизация и программирование на Turbo Pascal: от простых до олимпиадных задач. – СПб.: Питер, 2005. – 237 с.
7. Долинский, М.С. Решение сложных и олимпиадных задач по программированию. – СПб.: Питер, 2006. – 336 с.
8. Кирюхин, В.М. Информатика: всероссийские олимпиады. Выпуск 1. – М.: Просвещение, 2008. – 220 с.
9. Кирюхин, В.М., Окулов, С.М. Методика решения задач по информатике. Международные олимпиады. – М.: БИНОМ. Лаборатория знаний, 2007. – 600 с.
10. Меньшиков, Ф.В. Олимпиадные задачи по программированию. – СПб.: Питер, 2006. – 315 с.
11. Московские олимпиады по информатике / Под ред. Е.В. Андреевой, В.М. Гуровица и В.А. Матюхина. – М.: МЦНМО, 2006. – 256 с.
12. Московские учебно-тренировочные сборы по информатике. Весна-2006 / Под ред. В.М. Гуровица. – М.: МЦНМО, 2007. – 194 с.
13. Окулов, С.М. Программирование в алгоритмах. – М.: БИНОМ. Лаборатория знаний, 2002. – 341 с.
14. Окулов, С.М. и др. Информатика в задачах. – Киров: изд-во ВГПУ, 1998. – 343 с.
15. Оршанский, С.А. О решении олимпиадных задач по программированию в формате ACM ICPC. – Мир ПК, 2005, № 9.
16. Порублев, И.Н., Ставровский А.Б. Алгоритмы и программы. Решение олимпиадных задач. – М.: ООО «И.Д. Вильямс», 2007. – 480 с.
17. Скиена, С.С., Ревилла, М.А. Олимпиадные задачи по программированию. Руководство по подготовке к соревнованиям. – М.: КУДИЦ-ОБРАЗ, 2005. – 416 с.
18. Четвертая Всероссийская командная олимпиада школьников по программированию / Под ред. В.Н. Васильева, В.Г. Парфенова, А.С. Станкевича. – СПб.: СПбГУ ИТМО, 2003. – 155 с.
19. Шень, А.Х. Программирование: теоремы и задачи. – М.: МЦНМО, 2004. – 296 с.

Алексеев Александр Владимирович
Беляев Сергей Николаевич

# Подготовка школьников к олимпиадам по информатике с использованием веб-сайта

**Учебно-методическое пособие
для учащихся 7-11 классов**

***Оригинал-макет изготовлен РИО ИРО***

Технический редактор:
Сафронова Е.В.
Дизайн обложки:
Белов М.В.

Подписано в печать 05.06.2008.
Формат 60*84/16. Гарнитура Times New Roman.
Усл.п.л. 17,75. Заказ № 100. Тираж 500 экз.

**Институт развития образования**

Ханты-Мансийский автономный округ – Югра
628012, г. Ханты-Мансийск, ул. Чехова, 12