Администрация города Нижнего Новгорода
Департамент образования и социально-правовой защиты детства
Нижегородский государственный университет им. Н. И. Лобачевского

Шестая нижегородская городская олимпиада школьников по информатике

4 февраля 2010 г.

Нижний Новгород
2010

Результаты, архивы и другие материалы олимпиады
можно найти на сайте `http://olympiads.nnov.ru`

Оригинал-макет подготовлен в системе LaTeX 2$_\varepsilon$
с использованием набора шрифтов LH.

Шестая городская олимпиада по информатике

4 февраля 2010 г. департаментом образования и социально-правовой защиты детства администрации города Нижнего Новгорода и Нижегородским государственным университетом им. Н. И. Лобачевского, при активном участии Научно-образовательного центра ИПФ РАН и МОУ Лицей № 40 проводится шестая городская олимпиада по информатике среди учащихся 6-11 классов образовательных учреждений города Нижнего Новгорода. Целью проведения олимпиады является поиск талантливой молодёжи, привлечение её в науку, повышения уровня преподавания предметов физико-математического цикла.

Традиционно для участия в городской олимпиаде приглашаются талантливые школьники из Нижегородской области (г. Балахна, г. Саров).

Генеральным спонсором городской олимпиады школьников по информатике является компания «Мера НН».

Шестая городская олимпиада проводится на четырёх компьютерных площадках:

- Нижегородский государственный университет им. Н. И. Лобачевского (компьютерные классы механико-математического факультета) — 25 мест;
- Институт прикладной физики Российской академии наук (компьютерные классы Научно-образовательного центра) — 23 места;
- Муниципальное образовательное учреждение лицей № 40 (компьютерный центр) — 22 места;
- Муниципальное образовательное учреждение школа № 30 — 10 мест.

Олимпиада проводится в соответствии с Положением о городской олимпиаде по информатике (Приказ № 1453 от 07.11.2008 Департамента образования и СПЗД администрации г. Нижнего Новгорода).

Разрешённые языки программирования — Borland Pascal 7.0, Borland C 3.1, Borland C++ 3.1, Microsoft Basic 7.10. Ввод и вывод данных в программах осуществляется через файлы.

Регламент проведения олимпиады

9:30—10:00	Регистрация участников на всех компьютерных площадках одновременно (ННГУ, НОЦ ИПФ РАН, лицей № 40, школа № 30)
10:00—15:00	Решение задач олимпиады
15:00—16:00	Перерыв для сбора всех участников олимпиады в конференц-зале НОЦ ИПФ РАН, переезд в ИПФ РАН.
16:00—17:00	Открытое тестирование
17:00—18:00	Приветствие участников олимпиады, выступления учредителей, спонсоров. Подведение итогов олимпиады. Поздравление победителей и призёров.

Состав оргкомитета олимпиады

Швецов В. И., проректор по информатизации ННГУ им. Н. И. Лобачевского, профессор;
Сидоркина С. Л., первый заместитель директора департамента образования и социально-правовой защиты детства администрации города Нижнего Новгорода;
Лелюх В. Д., старший преподаватель ННГУ;
Цветков М. И., начальник отдела общего среднего образования департамента образования и социально-правовой защиты детства администрации города Нижнего Новгорода;
Бовкун И. Л., главный специалист отдела общего среднего образования департамента образования и социально-правовой защиты детства администрации города Нижнего Новгорода;
Смирнов А. И., директор НОЦ ИПФ РАН, профессор;
Фейгина Т. А., заместитель директора НОЦ ИПФ РАН;
Умнова Н. С., директор муниципального образовательного учреждения лицей № 40;
Антонова Л. Л., директор муниципального образовательного учреждения школа № 30;
Евстратова Л. П., заместитель директора по информатизации МОУ лицей № 40;
Гашпар И. Л., куратор НОЦ ИПФ РАН;
Братчикова Т. А., учитель МОУ лицей № 40;
Герасимова И. Н., учитель МОУ школа № 30;
Денисов В. В., зав. лабораторией ННГУ;

Состав предметной комиссии (жюри)

Председатель — Лелюх В. Д., старший преподаватель ННГУ;

Члены комиссии:
Кетков Ю. Л., профессор ННГУ;
Евстратова Л. П., заместитель директора МОУ лицей № 40, ответственный секретарь комиссии;
Бударагин Д. И., студент 4 курса ВШОПФ ННГУ;
Вадимов В. Л., студент 1 курса ВШОПФ ННГУ;
Епифанов В. Ю., студент 1 курса мехмата ННГУ;
Калинин П. А., аспирант ИПФ РАН;
Круглов А. А., младший научный сотрудник ИПФ РАН;
Люльков А. Е., студент 1 курса ВМК ННГУ;
Матросов М. В., студент 4 курса ВМК ННГУ;
Низовцев С. Н., студент 1 курса ВМК МГУ;
Разенштейн И. П., студент 3 курса мехмата МГУ;
Тимушев Р. И., старший разработчик Luxoft/UBS;
Тихонов А. О., студент 1 курса ВМК МГУ;
Хаймович И. М., аспирант ИФМ РАН;
Шмелёв А. С., студент 5 курса ВМК ННГУ.

Ниже приведены примеры текстов программ, написанных на разрешённых языках программирования. Программы считывают данные (два числа в одной строке) из файла с именем `example.in` и выводят в файл с именем `example.out` их сумму:

Pascal

```
var buf,bufo:text;
    a,b:integer;
begin
  assign(buf,'example.in');
  reset(buf);
  read(buf,a,b);
  assign(bufo,'example.out');
  rewrite(bufo);
  writeln(bufo,a+b);
  close(buf);
  close(bufo);
end.
```

C/C++

```
#include <stdio.h>
int main() {
  FILE *buf,*bufo;
  int a,b;
  buf=fopen("example.in","r");
  fscanf(buf,"%d %d",&a,&b);
  fclose(buf);
  bufo=fopen("example.out","w");
  fprintf(bufo,"%d\n",a+b);
  fclose(bufo);
  return 0;
}
```

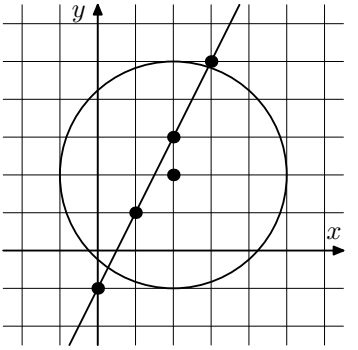
Basic

```
OPEN "example.in" FOR INPUT AS #1
OPEN "example.out" FOR OUTPUT AS #2
INPUT #1, A, B
PRINT #2, A + B
CLOSE #2, #1
```

VI Городская олимпиада школьников по информатике 4 февраля 2010 г.

Задача 1. Метро

Входной файл	`metro.in`
Выходной файл	`metro.out`
Ограничение по времени	1 с
Максимальный балл за задачу	100



В городе N строят метро. Вася, житель города N, хочет знать, сколько станций окажутся недалеко от его дома. Помогите ему.

Город N отличается очень строгой планировкой улиц: каждая улица идёт либо строго с юга на север, либо строго с востока на запад; при этом расстояние между соседними параллельными улицами одинаково. Соответственно, в городе есть много перекрёстков, расположенных в вершинах квадратной сетки. По планам, первая линия метро будет прямой и будет иметь станции на каждом перекрёстке, через который она пройдёт. Вася считает, что станция находится недалеко от его дома, если расстояние по прямой от его дома до станции не превосходит некоторой фиксированной величины R.

Формат входных данных

Введём систему координат с осью x, направленной с востока на запад, и осью y, направленной с юга на север, с началом координат на одном из перекрёстков и с единицей длины, равной расстоянию между соседними параллельными улицами. Таким образом, улицы будут прямыми с уравнениями $\ldots$, $x = -2$, $x = -1$, $x = 0$, $x = 1$, $x = 2$, $\ldots$, а также $\ldots$, $y = -2$, $y = -1$, $y = 0$, $y = 1$, $y = 2$, $\ldots$

Во первой строке входного файла находятся целые числа x_0, y_0 — координаты Васиного дома (считаем, что он находится на некотором перекрёстке), — и расстояние R в тех же единицах измерения, в которых введены координаты. Во второй строке находятся четыре числа x_1, y_1, x_2, y_2 — координаты некоторых двух различных перекрёстков, через которые пройдёт линия метро. Все координаты во входном файле не превосходят 100 000 000 по модулю; расстояние R целое, положительное и не превосходит 100 000 000.

Можете считать, что линия метро будет бесконечной в обоих направлениях.

Формат выходных данных

В выходной файл выведите одно число — количество станций, расположенных недалеко от Васиного дома.

Пример

Входной файл	*Выходной файл*
2 2 3 0 -1 1 1	2
0 0 1 -5 0 -3 0	3

Примечание: Первый пример соответствует рисунку; на рисунке дом Васи и станции метро обозначены жирными точками.

Решение

Фактически, в задаче требовалось по данным прямой и кругу определить, сколько на этой прямой есть точек с целочисленными координатами, лежащих внутри этого круга. Задача достаточно легко решалась следующим образом.

Нам известны две точки («станции») с целочисленными координатами ($P(x_1, y_1)$ и $Q(x_2, y_2)$), лежащие на прямой, но это не обязательно две *ближайшие* станции. Найдём расстояние между *ближайшими* станциями, а точнее, координаты вектора, соединяющего две ближайшие станции. Этот вектор должен иметь минимальные целочисленные координаты и быть пропорционален вектору $\overrightarrow{PQ}$; несложно видеть, что его координаты имеют вид

$$\begin{cases} \Delta x = \dfrac{x_2 - x_1}{d} \\ \Delta y = \dfrac{y_2 - y_1}{d}, \end{cases}$$

где $d = \text{НОД}(x_2 - x_1, y_2 - y_1)$ — наибольший общий делитель координат вектора $\overrightarrow{PQ}$.

Теперь несложно видеть, что при любом целом t формулы

$$\begin{cases} x = x_1 + t \cdot \Delta x \\ y = y_1 + t \cdot \Delta y, \end{cases}$$

определяют координаты некоторой станции; и наоборот, координаты *любой* станции получаются из этих формул при некотором целом t (в частности, при $t = 0$ получаем координаты точки P, а при $t = d$ — координаты точки Q).

Нам требуется определить, сколько из этих точек лежат внутри круга. Точка лежит внутри круга, если для нее выполняется условие

$$(x - x_0)^2 + (y - y_0)^2 \leqslant R^2.$$

Подставляя сюда выражения для x и y и раскрывая скобки, получаем квадратичное неравенство на t вида

$$at^2 + bt + c \leqslant 0,$$

где коэффициенты a, b и c легко выражаются через x_1, y_1, Δx, Δy, x_0, y_0 и R; в частности, отметим, что всегда $a = (\Delta x)^2 + (\Delta y)^2 > 0$.

Это неравенство либо не имеет решений (при $b^2 - 4ac < 0$), либо имеет решения

$$t_1 \leqslant t \leqslant t_2,$$

где t_1 и t_2 — корни квадратного уравнения $at^2 + bt + c = 0$ (с условием $t_1 \leqslant t_2$; в случае $b^2 - 4ac = 0$ получим $t_1 = t_2$). С учетом того, что t должно быть целое, получим

$$\lceil t_1 \rceil \leqslant t \leqslant \lfloor t_2 \rfloor,$$

где $\lceil x \rceil$ — округление x до ближайшего целого вверх , а $\lfloor x \rfloor$ — вниз.

Отсюда уже очевидно, что ответ на задачу равен

$$\lfloor t_2 \rfloor - \lceil t_1 \rceil + 1;$$

отметим, что эта формула корректна также и в случае, когда отрезок $[t_1, t_2]$ не включает ни одного целого числа (и ответ на задачу — ноль), и при $t_1 = t_2$.

Таким образом, для решения задачи достаточно реализовать предложенный алгоритм. Отметим также, что особое внимание надо уделить аккуратной работе с вещественными числами с учетом того, что при всех операциях накапливается погрешность; в частности, вместо округления лучше реализовать поиск ближайшей целочисленной точки, лежащей в круге, путем перебора нескольких соседних целочисленных точек и использования только целочисленной арифметики. Еще отметим, что полезно перенести начало координат в точку (x_0, y_0) (т.е. вычесть x_0 и y_0 из считанных из входного файла координат), что позволит упростить вид коэффициентов a, b и c.

Пример правильной программы

```
{$N+}
const eps=1e-5;
var f:text;
    x0,y0,r:longint;
    x1,y1,x2,y2:longint;
    l:longint;
    vx,vy:longint;
    a,b,c:extended;
    d:extended;
    t1,t2:extended;
    n1,n2:longint;

function incircle(t:comp):boolean;
var x,y:comp;
    rr:comp;
begin
  x:=x1+vx*t;
  y:=y1+vy*t;
  rr:=r;
  incircle:=x*x+y*y-rr*rr<=0;
end;

function gcd(a,b:longint):longint;
begin
  if b=0 then gcd:=a
  else gcd:=gcd(b,a mod b);
end;

function ceiltocircle(t:extended):longint;
var r,d:longint;
begin
  r:=round(t);d:=-5;
  while (not incircle(r+d))and(d<5) do
    inc(d);
  ceiltocircle:=r+d;
end;


function floortocircle(t:extended):longint;
var r,d:longint;
begin
  r:=round(t);d:=5;
  while (not incircle(r+d))and(d>-5) do
    dec(d);
  floortocircle:=r+d;
end;

begin
  assign(f,'metro.in');reset(f);
  read(f,x0,y0,r,x1,y1,x2,y2);
  x1:=x1-x0;
  y1:=y1-y0;
  x2:=x2-x0;
  y2:=y2-y0;
  l:=gcd(x2-x1,y2-y1);
  vx:=(x2-x1) div l;
  vy:=(y2-y1) div l;
  a:=1.0*vx*vx+1.0*vy*vy;
  b:=2.0*x1*vx+2.0*y1*vy;
  c:=1.0*x1*x1+1.0*y1*y1-1.0*R*R;
  d:=b*b-4*a*c;
  if d<-eps then begin
    assign(f,'metro.out');rewrite(f);
    writeln(f,0);
    close(f);
    halt;
  end;
  if d<0 then d:=0;
  t1:=(-b-sqrt(d))/2/a;
  t2:=(-b+sqrt(d))/2/a;
  n1:=ceiltocircle(t1);
  n2:=floortocircle(t2);
  assign(f,'metro.out');rewrite(f);
  if n1<=n2 then  writeln(f,n2-n1+1)
  else writeln(f,0);close(f);
end.
```

Задача 2. Автомат со сдачей

Входной файл	`change.in`
Выходной файл	`change.out`
Ограничение по времени	1 с
Максимальный балл за задачу	100

Фирма, в которой всё ещё работает ваш друг, решила установить в своих маршрутках автоматы по продаже чая и кофе, чтобы во время поездок и, особенно, во время ожидания в пробках, пассажиры могли с толком провести время.

Стоимость стакана чая и кофе в автомате предполагается установить равной пяти рублям. Автоматы будут принимать монеты по 5 и 10 рублей, а также купюры в 10, 50 и 100 рублей. Когда пассажиру надо выдавать сдачу (т.е. когда пассажир бросил в автомат десятирублёвую монету или 10-, 50- или 100-рублёвую купюру), автомат выдаёт сдачу пятирублёвыми монетами; если же пассажир бросил в автомат пятирублёвую монету, то автомат её сохраняет и может использовать для сдачи следующим пассажирам.

Ясно, что, чтобы обеспечить возможность выдачи сдачи всем покупателям, может потребоваться изначально загрузить в автомат некоторое количество пятирублёвых монет. Сейчас на маршрутках фирмы проходят испытания с целью определить минимальное количество монет, которые надо загрузить в автомат перед выездом маршрутки в рейс. Вам дан протокол одного из таких испытаний: известен порядок, в котором пассажиры оплачивали свои покупки различными монетами и купюрами. Определите, какое минимальное количество пятирублёвых монет должно было изначально находиться в автомате, чтобы всем пассажирам хватило сдачи.

Формат входных данных

В первой строке входного файла находится одно натуральное число N — количество покупок в автомате, которые были совершены в ходе испытания ($1 \leqslant N \leqslant \leqslant 50\,000$). Во второй строке находятся N натуральных чисел, каждое из которых равно номиналу монеты или купюры, которую использовал очередной покупатель для оплаты; каждый номинал может принимать одно из четырёх значений: 5, 10, 50 или 100.

Формат выходных данных

В выходной файл выведите одно число — минимальное количество пятирублёвых монет, которые надо было загрузить в автомат изначально, чтобы всем покупателям хватило сдачи.

Пример

Входной файл	*Выходной файл*
3 10 5 100	19
3 5 5 10	0
4 50 5 5 5	9

Примечание: В первом примере одна пятирублёвая монета потребуется для сдачи первому покупателю и 19 монет — третьему, но при сдаче третьему можно будет использовать ту монету, которую бросит второй покупатель, поэтому изначально в автомате достаточно 19 монет.

Во втором примере сдачу третьему покупателю можно выдать, используя монету первого или второго покупателя, и поэтому не требуется загружать монеты в автомат изначально.

В третьем примере первому же покупателю требуются девять монет сдачи, и все они должны изначально находится в автомате.

Решение

Предположим, что в автомате было a монет до того, как к нему подошёл первый пассажир. После каждого покупателя число монет в автомате меняется в зависимости от того, сколько денег дал пассажир. Обозначим через δ_i, на сколько изменилось число пятирублёвых монет в автомате после покупки i-го пассажира: если он дал 5 рублей, то число монет увеличивается на одну (т. е. $\delta_i = 1$), а если он дал десяти-, пятидесяти- или сторублёвую купюру, то число монет в автомате уменьшается (нужно дать сдачу), и δ_i соответственно равно -1, -9 и -19.

Таким образом, число монет в автомате равно $a + \delta_1 + \cdots + \delta_k$ после покупки k-го пассажира. Чтобы автомат исправно функционировал, необходимо, чтобы такие суммы, $a + \sum\limits_{i=1}^{k} \delta_i$, были неотрицательными, то есть $a \geqslant -\sum\limits_{i=1}^{k} \delta_i$ для каждого $1 \leqslant k \leqslant N$. Таким образом, необходимо, чтобы исходное число монет в автомате a было больше, чем $\max\limits_k \left(-\sum\limits_{i=1}^{k} \delta_i \right) = -\min\limits_k \sum\limits_{i=1}^{k} \delta_i$.

Легко понять, что это условие также является и достаточным, поэтому минимальное число монет, которое должно находиться в автомате, равно $-\min\limits_k \sum\limits_{i=1}^{k} \delta_i$, и его легко вычислить по мере чтения входных данных (определяя δ_i по величине купюры или монеты покупателя, суммируя их и обновляя минимальное значение).

Отметим также, что может потребоваться особо обработать случай, когда полученное таким образом значение a отрицательно (что соответствует тому, что первые покупатели накидали в автомат столько монет, что их хватит на всю последующую сдачу и ещё останется). В этом случае надо вывести 0; это можно обработать особо, а можно просто изначально инициализировать переменную, в которой мы будем хранить текущий минимум, значением 0, что и сделано в примере программы.

Пример правильной программы

```
var current, coin, min, i, n: longint;
begin
  current:=0;  min:=0;
  assign(input,'change.in');
  reset(input);
  read(n);
  for i:=1 to n do begin
    read(coin);
    case coin of
      5: inc(current);
      10: dec(current);
      50: dec(current,9);
      100: dec(current,19);
    end;
    if current<min then min:=current;
  end;
  close(input);
  assign(output,'change.out');
  rewrite(output);
  write(-min);
  close(output);
end.
```

Задача 3. Логарифм

Входной файл	`lg.in`
Выходной файл	`lg.out`
Ограничение по времени	1 с
Максимальный балл за задачу	100

Рассмотрим два числа a и b. По ним можно однозначно определить такое целое k, что

$$b^k \leqslant a < b^{k+1};$$

это k мы будем называть целой частью логарифма a по основанию b.

Напишите программу, которая будет вычислять целую часть логарифма.

Формат входных данных

В первой строке входного файла записано одно целое число a ($1 \leqslant a \leqslant 10^{100}$) без ведущих нулей. Во второй строке входного файла записано целое число b ($2 \leqslant b \leqslant \leqslant 100$).

Формат выходных данных

В выходной файл выведите одно число — целую часть логарифма a по основанию b без ведущих нулей.

Пример

Входной файл	*Выходной файл*
12345678987654321 3	33
8 2	3
2 5	0

Решение

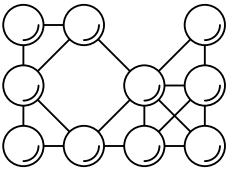
Будем решать задачу самым примитивным методом — возводить число b в степени 1, 2, ..., k, пока наше число b^k не станет больше a. Тогда очевидно, что $k-1$ и есть ответ на нашу задачу. Но не очевидно то, что наш алгоритм достаточно быстро найдёт решение и уложится в ограничение по времени. Попробуем оценить результат сверху. Для заданного b однозначно найдётся такое целое число x, что $b^{x-1} \leqslant 10 < b^x$. Заметим, что a всегда строго меньше, чем 10^n, где n — длина числа a в десятичной записи. Тогда $a < 10^n < b^{nx}$. Следовательно, искомое число k не больше nx. При наибольшем допустимом значении a, равном 10^{100}, значение n максимально и равно 101. Максимальное возможное значение x достигается при $b = 2$, когда $x = 4$. Поэтому для данных ограничений ответ всегда меньше, чем 404. Поэтому несложно видеть, что приведённое решение вполне укладывается в ограничение по времени. Для того, чтобы полностью решить задачу, необходимо реализовать длинную арифметику. В данной задаче нужны всего лишь две операции — умножение «длинного» числа на «короткое» и сравнение двух «длинных» чисел.

Пример правильной программы

```
const maxp=100;
      maxl=maxp+4;
type tlong=array[1..maxl] of longint;
var f:text;
    a,t:tlong;
    b:integer;
    pow:integer;

procedure readlong(var a:tlong);
var t,i,j:integer;
    ch:char;
begin
  i:=0;
  repeat
    read(f,ch);
    if not (ch in ['0'..'9']) then break;
    inc(i);
    a[i]:=ord(ch)-48;
  until false ;
  for j:=1 to i div 2 do begin
    t:=a[j];
    a[j]:=a[i+1-j];
    a[i+1-j]:=t;
  end;
end;

function lesseq(var a,b:tlong):boolean;
var i:integer;
begin
  for i:=maxl downto 1 do begin
    if a[i]<b[i] then begin
      lesseq:=true; exit;
    end;
    if a[i]>b[i] then begin
      lesseq:=false; exit;
    end;
  end;
  lesseq:=true;
end;

procedure mullong(var a:tlong; b:integer);
var i:integer;
begin
  for i:=1 to maxl do
    a[i]:=a[i]*b;
  for i:=1 to maxl-1 do begin
    a[i+1]:=a[i+1]+a[i] div 10;
    a[i]:=a[i] mod 10;
  end;
end;

begin
  assign(f,'lg.in');reset(f);
  readlong(a);
  read(f,b);
  close(f);
  fillchar(t,sizeof(t),0);
  t[1]:=1;
  pow:=0;
  while lesseq(t,a) do begin
    mullong(t,b);
    inc(pow);
  end;
  dec(pow);
  assign(f,'lg.out');rewrite(f);
  writeln(f,pow);
  close(f);
end.
```

Задача 4. Шарики

Входной файл	balls.in
Выходной файл	balls.out
Ограничение по времени	1 с
Максимальный балл за задачу	100



Фирма, в которой работает ваш друг, решила установить на конечной остановке своих маршруток большую абстрактную скульптуру со своим логотипом. Скульптура будет представлять собой прямоугольную сетку из N строк и M столбцов, в некоторых узлах которой будут располагаться разноцветные шары. Для обеспечения жёсткости конструкции шары, расположенные в узлах, соседних по вертикали, горизонтали или диагонали, необходимо соединить металлическими стержнями. Более строго, два шара должны быть соединены стержнем, если разность номеров строк, в которых расположены эти шары, не превосходит по модулю единицы, и разность номеров столбцов тоже не превосходит по модулю единицы.

Напишите программу, которая по заданному расположению шаров позволит определить, как будет выглядеть скульптура после установки стержней.

Формат входных данных

В первой строке входного файла находятся два натуральных числа N и M — размеры конструкции ($1 \leqslant N, M \leqslant 100$). Далее следуют N строк по M символов в каждой. Каждый символ — это или '`#`', обозначающий, что в соответствующем узле будет находиться шарик, или '`.`', обозначающий, что соответствующий узел будет пустой.

Формат выходных данных

Выведите в выходной файл $2N - 1$ строку по $2M - 1$ символов в каждой, изображающие как сами шары, так и соединяющие их стержни. А именно, в нечётных позициях нечётных строк выведите символ '`#`' или '`.`', в зависимости от того, заполнен этот узел шариком или нет, а в остальных позициях выведите один из символов ' ' (пробел), '`-`' (минус), '`|`' (вертикальная палочка, ASCII #124), '`/`' (дробь, ASCII #47), '\' (обратный слеш, ASCII #92) или '`X`' (латинская заглавная буква X, ASCII #88), отражающий конфигурацию стержней в соответствующем месте структуры.

Если какая-то строка выходного файла должна заканчиваться на пробелы, их можно не выводить.

Пример

Входной файл	*Выходной файл*
3 4 ##.# #.## ####	#-# . # |/ \ /| # . #-# |\ /|X| #-#-#-#
3 4 .#.. #.#. .#..	. # . . / \ # . # . \ / . # . .

Примечание: Первый пример соответствует рисунку.

Решение

Задача решалась легко, необходимо было лишь аккуратно реализовать требуемую логику. Можно было сохранить в памяти матрицу размера $(2N-1) \times (2M-1)$ и аккуратно её заполнить, а можно было даже считывать строки входного файла по одной и, храня в памяти только текущую и предыдущую строки, выводить данные в выходной файл. Последний вариант и реализован в примере решения.

Пример правильной программы

```
const maxm=100;
type line=array[1..maxm] of char;
var a1,a2:line;
    n,m,i,j:integer;

procedure draw_line(var l:line);
var i:integer;
begin
  write(l[1]);
  for i:=2 to m do begin
    if (l[i-1]='#') and (l[i]='#') then
      write('-')
    else write(' ');
    write(l[i]);
```

```
  end;
  writeln;
end;

procedure draw_between_lines(var l1,l2:line);
var i:integer;
begin
  if (l1[1]='#') and (l2[1]='#') then
    write('|')
  else write(' ');
  for i:=2 to m do begin
    if (l1[i-1]='#') and (l2[i-1]='#') and
              (l1[i]='#') and (l2[i]='#') then
      write('X')
    else if (l1[i-1]='#') and (l2[i]='#') then
      write('\')
    else if (l2[i-1]='#') and (l1[i]='#') then
      write('/')
    else
      write(' ');

    if (l1[i]='#') and (l2[i]='#') then
      write('|')
    else
      write(' ');
  end;
  writeln;
end;

begin
  assign(input,'balls.in');
  reset(input);
  assign(output,'balls.out');
  rewrite(output);
  readln(n,m);
  for i:=1 to m do read(a2[i]);
  readln;
  draw_line(a2);
  for i:=2 to n do begin
    a1:=a2;
    for j:=1 to m do read(a2[j]);
    readln;
    draw_between_lines(a1,a2);
    draw_line(a2);
  end;
  close(input);
  close(output);
end.
```

Задача 5. Совершенно секретно

Входной файл	secret.in
Выходной файл	secret.out
Ограничение по времени	1 с
Максимальный балл за задачу	100

В секретном 240 отделе ИПФ РАН N сотрудников и N компьютеров. В отделе вводятся новые требования к секретности. В соответствии с этими требованиями, для каждого сотрудника будут определены ровно K компьютеров, к которым этот сотрудник будет иметь допуск (т. е. за которыми этот сотрудник будет иметь право работать), причём так, что к каждому компьютеру будут иметь допуск ровно K сотрудников.

Информация о том, какой сотрудник к какому компьютеру будет иметь допуск, будет известна лишь непосредственно перед вступлением новых требований в силу. Таким образом, чтобы не прерывать работу отдела, сотрудники должны будут быстро решить, кто за каким компьютером будет работать. Для этого им необходимо заранее написать программу, которая по любому распределению допусков сотрудников найдёт рассадку сотрудников по компьютерам, удовлетворяющую этим допускам.

Обратите внимание, что значение K тоже будет известно лишь в последний момент. Из общих соображений секретности известно лишь, что K будет равняться или 1, или 2, или 4; поэтому ваша программа должна уметь работать для любого из этих трех значений K.

Формат входных данных

В первой строке входного файла записаны натуральные числа N и K ($1 \leqslant N \leqslant \leqslant 500$). Далее следуют KN строк, в каждой из которых записаны два натуральных

числа — номер сотрудника и номер компьютера, к которому этот сотрудник имеет допуск.

Гарантируется, что каждый сотрудник имеет допуск ровно к K компьютерам, что к каждому компьютеру ровно K сотрудников имеют допуск, и что K равно либо 1, либо 2, либо 4.

Формат выходных данных

В выходной файл выведите N строк, в каждой по два числа — номер сотрудника и номер компьютера, за которым он должен работать. Строки можно выводить в произвольном порядке.

Если есть несколько решений, выведите любое. Можно доказать, что для любого входного файла, удовлетворяющего указанным ограничениям, всегда имеется как минимум одно решение.

Пример

Входной файл	*Выходной файл*
3 1 2 3 3 1 1 2	3 1 1 2 2 3
2 2 1 2 2 1 2 2 1 1	1 2 2 1

Решение

Рассмотрим двудольный граф, в котором одной доле сопоставим работников, а другой — компьютеры. Ребро между вершиной i из первой доли и вершиной j из второй будет обозначать, что работник с номером i имеет допуск к компьютеру с номером j. Требуется выбрать для каждой вершины из первой доли вершину из второй, причём разным вершинам из первой доли должны соответствовать разные вершины из второй доли.

Случай, когда K равно 1: каждый работник имеет допуск ровно к одному компьютеру, и только он имеет допуск к этому компьютеру. Всё очевидно, и требуемая рассадка определятся однозначно.

Случай, когда K равно 2: возьмём любую компоненту связности в этом двудольном графе. Степень каждой вершины в этой компоненте чётна, а значит, в этой компоненте существует эйлеров цикл (цикл, в котором каждое ребро проходится ровно один раз). Удалим из нашего графа каждое второе ребро в этом цикле. После такой операции степень каждой вершины станет равна 1 (ребра из каждой вершины в цикле являются соседними, и одно из них мы оставим, а другое удалим). Проведя эту операцию с каждой компонентой связности, мы перейдём к уже рассмотренному случаю $K = 1$.

Случай, когда K равно 4, решается аналогично предыдущему, с той лишь разницей, что описанную операцию надо будет произвести дважды: степень каждой вершины после первого удаления станет равной 2, а после второго — 1.

Отметим также, что для случая $K = 2$ существует идейно более простое решение. А именно, несложно показать, что в графе, в котором степень каждой вершины равна 2, каждая компонента связности — это цикл. Действительно, встанем в произвольную вершину и пойдём из неё по какому-нибудь выходящему из неё ребру. Из той вершины, куда мы придём, выходят ровно два ребра, по одному мы пришли — по другому пойдём далее, и т.д. Придя в очередную вершину по одному ребру, пойдём из неё дальше по второму ребру, выходящему из этой вершины. Т. к. граф конечен, мы когда-нибудь вернёмся в вершину, в которой уже были, и несложно видеть, что это будет та вершина, из которой мы начали обход. Таким образом, мы обойдем всю соответствующую компоненту связности, получив цикл, и, если в этом цикле оставить ребра через одно, то получится корректное решение задачи в этой компоненте связности. Повторив это для всех компонент связности, мы решим задачу.

Пример правильной программы

```
{$M 65520,0,655360}
const max=1000;
type pedge=^tedge;
     tedge=record
             v:integer;
             next:pedge;
             pair:pedge;
             used:boolean;
           end;
var f:text;
    gr,grnew:array[1..max] of pedge;
    d:array[1..max] of integer;
    dd:integer;
    n,m,k:integer;
    nc:integer;
    i:integer;
    u,v:integer;
    p:pedge;
    was:array[1..max] of integer;
    bipart:boolean;
    prev:integer;

procedure add(u,v:integer);
var p1,p2:pedge;
begin
  new(p1);
  p1^.v:=v;
  p1^.next:=grnew[u];
  p1^.used:=false;
  grnew[u]:=p1;

  new(p2);
  p2^.v:=u;
  p2^.next:=grnew[v];
  p2^.used:=false;
  grnew[v]:=p2;
  p1^.pair:=p2;
  p2^.pair:=p1;
end;

procedure find(i:integer);
var p:pedge;
    q:pedge;
    v:integer;
begin
  was[i]:=1;
  while gr[i]<>nil do begin
    if not gr[i]^.used then begin
      gr[i]^.used:=true;
      gr[i]^.pair^.used:=true;
      v:=gr[i]^.v;
      gr[i]:=gr[i]^.next;
      find(v);
    end
    else gr[i]:=gr[i]^.next;
  end;
  inc(nc);
  if nc and 1=0 then
    add(prev,i);
  prev:=i;
end;

begin
  assign(f,'secret.in');reset(f);
  read(f,n,k);
  m:=k*n;
  n:=n*2;
  fillchar(grnew,sizeof(grnew),0);
  for i:=1 to m do begin
    read(f,u,v);
    v:=v+n div 2;
    add(u,v);
    inc(d[u]);
```

```
    inc(d[v]);
  end;
  gr:=grnew;
  dd:=d[1];
  while dd>1 do begin
    nc:=-1;
    prev:=-1;
    fillchar(grnew,sizeof(grnew),0);
    fillchar(was,sizeof(was),0);
    for i:=1 to n do
      if was[i]=0 then begin
        inc(nc);
        find(i);
      end;
    gr:=grnew;
    dd:=dd div 2;
  end;
  assign(f,'secret.out');rewrite(f);
  for i:=1 to n div 2 do begin
    p:=gr[i];
    u:=p^.v;
    writeln(f,i,' ',u-n div 2);
  end;
  close(f);
end.
```

Задача 6. Шифр

Входной файл	`cypher.in`
Выходной файл	`cypher.out`
Ограничение по времени	1 с
Максимальный балл за задачу	100

Жюри N-ской олимпиады по информатике решило зашифровать свои материалы подстановочным шифром. Для шифрования таким шифром задаётся взаимнооднозначное соответствие между буквами алфавита в открытом (т. е. до шифрования) тексте и буквами алфавита в закрытом (т. е. после шифрования) тексте, это соответствие и является ключом шифра. В процессе шифрования каждая буква в открытом тексте заменяется на соответствующую ей букву в закрытом тексте, порядок букв в слове при этом не меняется.

Однако, память у членов жюри оказалась уже не та, что в молодые годы, поэтому часто они путали, какие буквы надо было заменять на какие. В результате теперь они не могут восстановить свои материалы, а олимпиада уже на носу!

Чтобы разрешить свои проблемы, они обратились к вам. Для облегчения вашей задачи они выписали на бумажку все возможные варианты зашифрования букв, которые они могли применять, в виде набора пар «открытая буква» — «зашифрованная буква». Также вам известны все пары букв N-ского алфавита, которые могут следовать одна за другой в открытом тексте. Ваша задача состоит в том, чтобы по заданному зашифрованному слову сказать, соответствует ли ему хоть одно расшифрованное слово, единственен ли вариант расшифровки, и привести пример вариантов расшифровки слова. Слово $\mathcal{A}$ считается возможной расшифровкой слова $\mathcal{B}$, если, во-первых, его можно «зашифровать» (заменяя каждую букву на одну из соответствующих ей «зашифрованных» букв), получив слово $\mathcal{B}$, и, во-вторых, каждая пара букв слова $\mathcal{A}$, стоящих рядом, является допустимой для N-ского языка.

Формат входных данных

N-ский язык пользуется латинским алфавитом из 26 букв, регистр букв N-ское жюри не интересует, поэтому везде в открытом тексте используются большие буквы, а в закрытом — маленькие.

На первой строке входного файла находится одно целое число M ($0 \leqslant M \leqslant 676$) — число пар открытых — «зашифрованных» букв, указанных на бумажке, переданной N-ским жюри. Далее следуют M строк, на каждой находятся два символа — сначала открытая буква, потом вариант её «зашифрования». Пары не повторяются.

На следующей строке находится одно целое число K ($0 \leqslant K \leqslant 676$) — число пар открытых букв, которые могут идти одна за другой. Далее следуют K строк, на каждой из которых по две открытые буквы, образующие такую пару. Пары не повторяются. Заметим, что возможна ситуация, когда последовательность букв '`AB`' в слове допустима, а '`BA`' — нет, в этом случае списке будет дана только пара '`AB`', а пары '`BA`' не будет.

На следующей строке расположено одно целое число N ($1 \leqslant N \leqslant 500$) — длина зашифрованного слова, а на следующей строке — само слово (N маленьких латинских букв).

Может оказаться так, что какой-то открытой букве не соответствует ни одна «зашифрованная»; это означает, что эта буква в открытом тексте не использовалась.

Формат выходных данных

Если вариантов дешифрования нет ни одного, в первую строку выходного файла выведите '`no`'.

Если вариант дешифрования ровно один, в первую строку выходного файла выведите '`only`', а во вторую — дешифрованное слово.

Если вариантов дешифрования больше одного, в первую строку выходного файла выведите '`many`', а во вторую и третью — любые два различных варианта дешифрования слова.

Пример

Входной файл	*Выходной файл*
1 Uz 0 2 zz	no
1 Uz 0 1 z	only U
2 Uz Xz 3 UU XX XU 2 zz	many UU XU

Входной файл	*Выходной файл*
7 Aa Az Ax Cz Dv Bx Bz 5 AB CA BC CB DE 4 zzxx	only BCAB

Решение

Задача легко решается методом динамического программирования. Обозначим через s зашифрованное слово, заданное во входном файле, пусть N — его длина. Для каждого числа i от 1 до N и для каждой буквы α от `A` до `Z` посчитаем $W[i, \alpha]$ — число заканчивающихся на букву α вариантов расшифровки слова $s[1..i]$ (т. е. подстроки, состоящей из первых i символов строки s). Во-первых, очевидно, что если i-й символ строки s ($s[i]$; символы нумеруем с 1) не может расшифровываться в букву α, то это число равно нулю. Пусть символ $s[i]$ может расшифровываться в букву α. Переберём все буквы β и посчитаем количество способов расшифровки строки $s[1..i]$, у которых в конце идёт строка $\beta\alpha$ (т. е. последний символ равен α, а предпоследний — β); очевидно, что искомое значение $W[i, \alpha]$ будет суммой по всем β посчитанных величин. Подсчёт же этого количества при данном β сложностей не вызывает: если буква α может идти за β в расшифрованном тексте, то ответ равен $W[i-1, \beta]$, иначе равен нулю.

Значения при $i = 1$ вычисляются очевидным образом: если символ $s[1]$ может перейти в α, то $W[1, \alpha] = 1$, иначе $W[1, \alpha] = 0$. Далее можно в двойном цикле по i и α вычислить все значения W. Таким образом мы сможем получить ответ на первую часть задачи (необходимо только заметить, что в процессе такого решения могут получаться очень большие числа, но, т. к. нас на самом деле интересует, равно ли каждое число нулю, единице, или же оно больше либо равно двух, то можно вместо всех значений, бóльших двух, хранить просто число два).

Вывод решения в задачах на динамическое программирование почти всегда осуществляется довольно просто по насчитанной матрице. Проиллюстрируем это на примере нашей задачи. Пусть для начала нам надо вывести только один любой вариант расшифровки. Для этого можно поступить следующим простым способом. Напишем процедуру $out(i, \alpha)$, которая будет выводить в выходной файл произвольную расшифровку строки $s[1..i]$, заканчивающихся на символ α (естественно, при условии, что $W[i, \alpha] \neq 0$). Если $i = 1$, то процедура просто выводит символ α и на этом заканчивает свою работу. Если же $i \neq 1$, то процедура, перебрав все буквы, находит такую букву β, что $W[i-1, \beta] \neq 0$ и α может следовать за β (такая буква найдётся, т. к. иначе $W[i, \alpha]$ было бы рано нулю), вызывает сама себя рекурсивно: $out(i-1, \beta)$ и после этого выводит символ α. Теперь, для вывода какого-нибудь решения основной задачи надо перебрать все буквы, найти такую α, что $W[N, \alpha] \neq 0$, и вызвать $out(N, \alpha)$.

Для вывода *двух* решений в случае необходимости можно поступить двумя способами: можно написать процедуру $out(i, \alpha, w)$, которая выводит w-е в каком-нибудь (например, алфавитном) порядке решение (естественно, при условии $1 \leqslant w \leqslant W[i, \alpha]$). Она перебирает все β, «пропуская» решения из $W[i, \alpha]$ и подсчитывая количество w' пропущенных решений (в точности как при насчитывании самого значения $W[i, \alpha]$, т. е. добавляя количество решений, заканчивающихся на $\beta\alpha$, к w'). Пусть при некотором β стало $w' \geqslant w$, причём при предыдущем β было $w' < w$ — обозначим это значение w' через w_0. Тогда очевидно, что искомое решение будет соответствовать $(w - w_0)$-му среди решений $W[i-1, \beta]$, поэтому вызываем $out(i-1, \beta, w - w_0)$, после чего выводим α.

Этот способ можно модифицировать, учтя, что нам надо считать только до двух,

и поэтому просто можно хранить, пропустили ли мы какое-нибудь решение или ещё пока нет. А именно, пусть мы, перебирая β, наткнулись на β, дающее несколько (т. е. не ноль) решений для наших i и α. Тогда: если $W[i-1,\beta] \geqslant 2$, то вызываем $out(i-1,\beta,w)$, иначе: если мы уже пропускали решения в этом вызове функции *out* **или** $w = 1$, то вызываем $out(i-1,\beta,1)$, иначе пропускаем это (единственное, т. к. $W[i-1,\beta] = 1$) решение. Этот способ и реализован в примере решения. В таком виде оно не всегда выводит первое и *второе* решения, но всегда первое и какое-то ещё.

Пример правильной программы

```
const MaxN=500;
      sss:array[0..2] of string=
                 ('no','only','many');
var f:text;
    can:array['a'..'z','A'..'Z'] of
                                boolean;
    can1:array['A'..'Z','A'..'Z'] of
                                boolean;
    nn:array[0..MaxN,'A'..'Z']of byte;
    s:array[1..MaxN] of char;
    ch,ch1,ch2:char;
    n:integer;
    i:integer;
    nall:integer;
    n1,n2:integer;

procedure out
        (i:integer;ch:char;n:integer);
var ch1:char;
    was:boolean;
begin
  if i>1 then begin
    was:=false;
    for ch1:='A' to 'Z' do
      if can1[ch1,ch] then
        case nn[i-1,ch1] of
          1: begin
            if was or (n=1) then begin
              out(i-1,ch1,1);
              break;
            end;
            was:=true;
          end;
          2: begin
            out(i-1,ch1,n);
            break;
          end;
        end;
  end;
  write(f,ch);
end;

begin
  assign(f,'cypher.in');reset(f);
  readln(f,n);
  fillchar(can,sizeof(can),0);
  for i:=1 to n do begin
    readln(f,ch2,ch1);
    can[ch1,ch2]:=true;
  end;
  readln(f,n);
  fillchar(can1,sizeof(can1),0);
  for i:=1 to n do begin
    readln(f,ch1,ch2);
    can1[ch1,ch2]:=true;
  end;
  readln(f,n);
  for i:=1 to n do read(f,s[i]);
  close(f);
  fillchar(nn,sizeof(nn),0);
  for ch1:='A' to 'Z' do
    if can[s[1],ch1] then nn[1,ch1]:=1;
  for i:=2 to n do begin
    for ch1:='A' to 'Z' do begin
      if can[s[i],ch1] then
        for ch2:='A' to 'Z' do
          if can1[ch2,ch1] then
            inc(nn[i,ch1],nn[i-1,ch2]);
      if nn[i,ch1]>2 then
        nn[i,ch1]:=2;
    end;
  end;
  assign(f,'cypher.out');rewrite(f);
  nall:=0;  n1:=1;n2:=1;
  for ch:='A' to 'Z' do begin
    if nn[n,ch]=2 then begin
      ch1:=ch;ch2:=ch;
      n2:=2;
    end;
    if nn[n,ch]=1 then begin
      if nall=0 then
        ch1:=ch
      else ch2:=ch;
    end;
    inc(nall,nn[n,ch]);
    if nall>=2 then
      break;
  end;
  if nall>2 then nall:=2;
  writeln(f,sss[nall]);
  if nall>=1 then
    out(n,ch1,n1);
  writeln(f);
  if nall>=2 then
    out(n,ch2,n2);
  writeln(f);
  close(f);
end.
```

Содержание

Шестая городская олимпиада по информатике 3
Регламент проведения олимпиады 4
Состав оргкомитета олимпиады 4
Состав предметной комиссии (жюри) 5
Задачи 6
1. Метро 6
2. Автомат со сдачей 9
3. Логарифм 11
4. Шарики 12
5. Совершенно секретно 14
6. Шифр 17