**A. Кондиціонер Степана**

|  |  |
| --- | --- |
| **Input file name:** | cond.in |
| **Output file name:** | cond.out |
| **Time limit:** | 100 ms |
| **Memory limit:** | 256 M |

В офісі, де Степан працює програмістом, встановили кондиціонер нового типу. Цей кондиціонер відрізняється особливою простотою в управлінні. У кондиціонера є всього лише два керованих параметра: бажана температура і режим роботи.

Кондиціонер може працювати в наступних чотирьох режимах:

- «freeze» - охолодження. У цьому режимі кондиціонер може тільки зменшувати температуру. Якщо температура в кімнаті і так не більше бажаної, то він вимикається.

- «heat» - нагрів. У цьому режимі кондиціонер може тільки збільшувати температуру. Якщо температура в кімнаті і так не менше бажаної, то він вимикається.

- «auto» - автоматичний режим. У цьому режимі кондиціонер може як збільшувати, так і зменшувати температуру в кімнаті до бажаної.

- «fan» - вентиляція. У цьому режимі кондиціонер здійснює тільки вентиляцію повітря і не змінює температуру в кімнаті.

Кондиціонер досить потужний, тому при налаштуванні на правильний режим роботи він за годину доводить температуру в кімнаті до бажаної.

Потрібно написати програму, яка по заданій температурі в кімнаті ***troom***, встановленим на кондиціонері бажаної температурі ***tcond*** і режиму роботи визначає температуру, яка встановиться в кімнаті через годину.

**Формат вхідних даних:** Перший рядок вхідного файлу містить два цілих числа ***troom***, і ***tcond***, розділених рівно одним пропуском ***(-50 ≤ troom ≤ 50 , -50 ≤ tcond ≤ 50)***.  
Другий рядок містить одне слово, записане малими літерами латинського алфавіту - режим роботи кондиціонера.

**Формат вихідних даних:** Вихідний файл повинен містити одне ціле число - температуру, яка встановиться в кімнаті через годину.

***Пояснення до прикладів:***

У першому прикладі кондиціонер знаходиться в режимі нагріву. Через годину він нагріє кімнату до бажаної температури в 20 градусів.

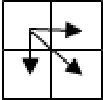
У другому прикладі кондиціонер знаходиться в режимі охолодження. Оскільки температура в кімнаті нижча, ніж бажана, кондиціонер самостійно вимикається і температура в кімнаті не поміняється.

**Приклади вхідних та вихідних даних:**

|  |  |
| --- | --- |
| **cond.in** | **cond.out** |
| 10 20  heat | 20 |
| 10 20  freeze | 10 |

**B. "Поле чудес"**

|  |  |
| --- | --- |
| **Input file name:** | wonderland.in |
| **Output file name:** | wonderland.out |
| **Time limit:** | 100 ms |
| **Memory limit:** | 256 M |

Степан пройшов до суперфіналу новорічного шоу "Поле чудес". На відміну від звичайних ігор новорічна відрізняється тим, що проводиться не за круглим барабаном, а на прямокутному полі, розбитому на квадратики. Кожен такий квадратик містить одне число (числа можуть бути як доданими, так і від'ємними). Гравець розташовується у верхній лівій клітинці і може переміщуватись у три сусідніх клітинки: праву, нижню та праву нижню по діагоналі (див. малюнок) і повинен потрапити до правої нижньої клітинки.

Звісно, що Степан має бажання виграти гру і набрати якомога більше балів. Якщо набрана сума додатна, то Степан виграв, інакше програв.

Напишіть програму, яка доможе Степану визначити програє чи виграє він у грі і яку суму він зможе набрати.

**Формат вхідних даних:** Перший рядок вхідного файлу містить два цілих числа ***N, M (1 ≤ N ≤ 100, 1 ≤ M ≤ 100)*** - розміри ігрового поля.

Наступні ***N*** рядків містять по ***М*** цілих чисел - значення клітинок ігрового поля, по модулю не більші за 1000.

**Формат вихідних даних:** Перший рядок вихідного файлу має містити одне слово - ***winner***, якщо Степан виграв гру, або ***loser*** - в іншому випадку.

Другий рядок має містити одне число - набрану суму.

**Приклади вхідних та вихідних даних:**

|  |  |
| --- | --- |
| **wonderland.in** | **wonderland.out** |
| 2 3  2 1 1  -1 2 1 | winner  6 |
| 2 3  1 -2 -1  -2 1 -2 | loser  0 |

**C. Winter**

|  |  |
| --- | --- |
| **Input file name:** | winter.in |
| **Output file name:** | winter.out |
| **Time limit:** | 500 ms |
| **Memory limit:** | 256 M |

Країна Ужляндія славиться своїми ідеальними дорогами, але навіть вони не витримали цьогорічної аномально холодної та сніжної зими. Деякі з доріг виявилися заблокованими для руху автомобілістів. Внаслідок цього порушився зв’язок між містами Ужляндії. Два міста країни вважаються з’єднаними, якщо можна дістатися з одного міста в інше, рухаючись не заблокованими дорогами, можливо, через інші міста.

Сусідня братська держава відома на весь світ своїми унікальними обігрівачами. Керівництво країни вирішило надати Ужляндії гуманітарну допомогу. Було вирішено, що обігрівачі доставлятимуться на гвинтокрилі, а далі за допомогою вантажівок розвозитимуться по містах. Оскільки авіаційне паливо не дешеве, потрібно мінімізувати кількість приземлень гвинтокрила так, щоби кожне місто отримало необхідні обігрівачі. Будь ласка, якомога швидше порахуйте цю кількість і врятуйте мешканців Ужляндії.

**Формат вхідних даних:** В першому рядку записано два числа ***N*** і ***M (1 ≤ N ≤ 100000,0 ≤ M ≤ 200000)*** – кількість міст в Ужляндії та кількість не заблокованих доріг відповідно. В наступних ***M*** рядках записано по два числа ***i*** та ***j (1 ≤ i, j ≤ N)***, що значить дорога між містами з номерами ***i*** та ***j*** не заблокована. Міста в Ужляндії нумеруються в 1 до ***N***.

**Формат вихідних даних:** В єдиному рядку виведіть мінімальну кількість приземлень гвинтокрила.

***Пояснення до прикладу:***

Міста 1, 2 та 3 з’єднані між собою, а тому щоби забезпечити їх обігрівачами, необхідно здійснити одне приземлення в одному з цих міст, далі обігрівачі доставлять вантажівками. Міста 4 та 5 зв’язані між собою, тому треба ще одне приземлення. І нарешті місто 6, яке ізольоване від інших, щоби доставити обігрівачі в це місто, треба окреме приземлення гвинтокрила. Всього виходить 3 приземлення.

**Приклади вхідних та вихідних даних:**

|  |  |
| --- | --- |
| **winter.in** | **winter.out** |
| 6 4  3 1  1 2  5 4  2 3 | 3 |

**D. Гарні числа**

|  |  |
| --- | --- |
| **Input file name:** | beautiful.in |
| **Output file name:** | beautiful.out |
| **Time limit:** | 1000 ms |
| **Memory limit:** | 256 M |

Школа №1331 в Ужляндії відома дуже високим рівнем знань своїх учнів з математики, тому що більшість учнів відвідують факультативні заняття відомого вчителя Антона Андрійовича.

Сьогодні Антон Андрійович розказав своїм учням про числа, які, на його думку, можуть володіти рядом цікавих властивостей. Він назвав такі числа гарними. Число називається гарним, якщо не існує такого цілого числа, більшого одиниці, на квадрат якого воно б ділилося без залишку. Наприклад, число 12 не є гарним, тому що воно ділиться на 4, тобто на квадрат числа 2. Числа 13 і 14 є гарними числами.

Учні Антона Андрійовича дуже гарні в усному рахунку, тому в першому завданні необхідно було визначити: чи є деяке число гарним.

Однак, Марися, краща його учениця, швидко впоралась з цим завданням. Щоб якось її зайняти, вчитель написав на дошці ***N*** чисел і дав їй нове завдання : визначити, чи є добуток цих чисел гарним числом. Дуже скоро Марися отримала відповідь, однак вона хоче перевірити себе. Тому вона просить Вас написати програму, яка перевіряє: чи є добуток чисел гарним числом, а якщо ні, то їй потрібно знати яке-небудь число, відмінне від одиниці, на квадрат якого ділиться добуток цих чисел.

**Формат вхідних даних:** Перший рядок вхідного файлу містить число ***N (1 ≤ N ≤ 100)*** - кількість чисел, які вчитель написав на дошці для Марисі. Другий рядок містить ***N*** натуральних чисел - самі числа. Кожне з чисел не перевершує 1018.

**Формат вихідних даних:** Якщо число є гарним, виведіть єдиний рядок, що складається з слова Beautiful. Інакше, виведіть яке-небудь число, відмінне від одиниці, на квадрат якого ділиться добуток ***N*** чисел.

***Пояснення до прикладів:***

Перший приклад: 5\*6\*7 = 210. Не існує числа, більшого за одиницю, на квадрат якого 210 ділилось би без залишку, тому 210 - гарне число.

Другий приклад: 35\*12 = 420. 420 ділиться на 4, тобто на квадрат числа 2.

***Оцінювання:***

***A1 \* A2 \* ... \* AN ≤ 106*** – не менше 20 балів

***A1 \* A2 \* ... \* AN ≤ 1012*** – не менше 40 балів

Для усіх ***і***: ***Ai ≤ 1012*** – не менше 60 балів

**Приклади вхідних та вихідних даних:**

|  |  |
| --- | --- |
| **beautiful.in** | **beautiful.out** |
| 3  5 6 7 | Beautiful |
| 2  35 12 | 2 |

**E. Вправи Степана**

|  |  |
| --- | --- |
| **Input file name:** | exercises.in |
| **Output file name:** | exercises.out |
| **Time limit:** | 1000 ms |
| **Memory limit:** | 128 M |

Степан вирішив досягти успіху не тільки в спортивному програмуванні, а й у спорті. На жаль, пройшло вже багато часу з дня його останнього тренування, тому, щоб набрати хорошу форму, доведеться починати з нуля.

Придумати вправи для тренувань виявилося непросто, тому Степан вирішив пошукати ідеї в Інтернеті. Він знайшов сайт, на якому пропонувалася кілька серій тренувальних вправ. Кожна серія тренувань за планом займає ***N*** днів. У кожен з цих ***N*** днів пропонується робити одну «вправу дня», а також до нього додаються рекомендації щодо виконання у вигляді ***"Ai - Bi"***. Це позначає, що для підвищення рівня сили потрібно виконувати вправу від ***Ai*** до ***Bi*** раз. Якщо виконувати вправу менш, ніж ***Ai***, або більш, ніж ***Bi*** раз, то це принесе скоріш за шкоду, ніж користь. Степан не хоче завдавати собі шкоди, тому буде робити від ***Ai*** до ***Bi*** раз, або зовсім не робити цю вправу.

Почитавши всі описи вправ, Степан зрозумів, що цей курс не розрахований на новачків, але вирішив не здаватися і адаптувати курс вправ під себе. Він знає, що при вивченні ***i***-ї вправи йому доведеться втратити ***Ki*** рівнів сили, зате за виконання вправи ***X*** раз його рівень сили зросте на ***Fi\*X***. Степан не може вивчити і виконати вправу, якщо його поточний рівень сили менший за ***Ki***. У дні, коли Степану не вистачає сил або часу тренуватись, він може пропускати тренування, і рівень його сили залишається без зміни. Знаючи свої можливості, Степан розуміє, що якщо в якийсь день він виконає вправу більш ***T*** разів, то наступні ***D*** днів він буде занадто втомленим, щоб займатися.

Якщо Степан виконає вправу більш ***T*** разів в якийсь з останніх ***T*** днів серії тренувань, то він почне відпочивати з наступного дня, а закінчить вже після кінця серії.  
Степан хоче отримати від занять максимум користі, тому він планує витратити на них ***N*** днів! Для кожної серії тренувань допоможіть йому визначити максимальної рівень сили, який він зможе досягти в кінці тренувань. До початку тренувань рівень сили Афанасія дорівнює нулю.

**Формат вхідних даних:** Перший рядок вхідного файлу містить єдине ціле число ***N (1 ≤ N ≤ 105)*** - кількість днів тренувань.

Другий рядок містить два цілих числа ***T, D (1 ≤ T ≤ 106, 1 ≤ D ≤ 105)***, якщо в якийсь день Степан виконає вправу більш ***T*** разів, то йому доведеться відпочивати ***D*** наступних днів.  
Наступні ***N*** рядків описують вправи, ***i+2***-ий рядок містить опис вправи в день ***i***. Кожна вправа описується числами ***Ai, Bi, Ki, Fi, (0 ≤ Ki ≤ 109, 1 ≤ Ai ≤ Bi ≤ 106, 1 ≤ Fi ≤ 106)***, розділеними одиночними пробілами, - де ***Ai, Bi*** відповідно рекомендований мінімум і максимум кількості разів виконання вправи, ***Ki***-кількість втрачаються рівнів сили при вивченні вправи, ***Fi***- кількість рівнів сили, одержуваних за кожен раз виконання вправи.

**Формат вихідних даних:** Перший рядок вихідного файлу має містити одне ціле число ***S*** - максимальний рівень сили, який Степан може досягти до кінця тренувань.  
Наступний рядок повинен містити ***N*** цілих чисел ***Xi*** - кількість разів виконання вправи в день ***і***, якщо в ***і***-ий день він відпочивав, то виведіть 0.

***Пояснення до прикладів:***

Щоб досягти максимального рівня сили, треба в перший день виконувати вправу 4 рази, щоб не довелося пропускати наступний день. У другий день Афанасій зможе збільшити свій рівень ще на 790 (втрачає 10 рівнів при вивченні та виконує вправу 8 разів), але тоді він не зможе займатися 1 день (третій день).

У четвертий день він збільшує свій рівень на 48, так як Степан виконав вправу більше 4 разів, він змушений пропустити п'ятий.

**Приклади вхідних та вихідних даних:**

|  |  |
| --- | --- |
| **exercises.in** | **exercises.out** |
| 5  4 1  3 5 0 10  6 8 10 100  2 8 10 15  5 6 0 8  2 2 1 7 | 878  4 8 0 6 0 |

#include<fstream>  
#include<string>  
using namespace std;  
ifstream in("cond.in");  
ofstream out("cond.out");  
string s;  
int a,b;  
int main()  
{  
        in>>a>>b;  
        in>>s;  
        if (s=="heat")  
        {  
                if (a<=b) out<<b<<endl; else out<<a<<endl;  
        }  
        else  
        {  
                if (s=="freeze")  
                {  
                        if (a>=b) out<<b<<endl; else out<<a<<endl;  
                }  
                else  
                {  
                        if (s=="auto") out<<b<<endl; else  
                                if (s=="fan") out<<a<<endl;  
                }  
        }  
}

#include<fstream>  
#include<string>  
using namespace std;  
ifstream in("wonderland.in");  
ofstream out("wonderland.out");  
int i,j,n,m,a[101][101],b[101][101],x,y;  
int main()  
{  
        in>>n>>m;  
        for (i=1;i<=n;i++)  
                for (j=1;j<=m;j++)  
                {  
                        in>>a[i][j];  
                        b[i][j]=-999999;  
                }  
        b[1][1]=a[1][1];  
        for (i=2;i<=n;i++)  
                b[i][1]=b[i-1][1]+a[i][1];  
        for (j=2;j<=m;j++)  
                b[1][j]=b[1][j-1]+a[1][j];  
        //for (i=2;i<=min(n,m);i++)  
        //        b[i][i]=b[i-1][i-1]+a[i][i];  
        i=2;j=2;  
        while (i!=min(n,m)+1 || j!=min(m,n)+1)  
        {  
                b[i][j]=max(max(b[i-1][j]+a[i][j],b[i][j]),max(b[i-1][j-1]+a[i][j],b[i][j-1]+a[i][j]));  
                for (x=2;x<=n;x++)  
                        b[x][j]=max(max(b[x-1][j]+a[x][j],b[x][j]),max(b[x-1][j-1]+a[x][j],b[x][j-1]+a[x][j]));  
                for (y=2;y<=m;y++)  
                        b[i][y]=max(max(b[i-1][y]+a[i][y],b[i][y]),max(b[i-1][y-1]+a[i][y],b[i][y-1]+a[i][y]));  
                i++;j++;  
        }  
        i=1;j=1;  
        if (b[n][m]>0) out<<"winner"<<endl; else out<<"loser"<<endl;  
        out<<b[n][m]<<endl;  
}

#include "stdafx.h"

#include "iostream"

using namespace std;

int a[100][100];

int main()

{int m,n,i,j;

cin>>n>>m;

for(i=1;i<=n;i++)

for(j=1;j<=m;j++)

cin>>a[i][j];

for(i=1;i<=n;i++)

for(j=1;j<=m;j++)

a[i][j]=max(max(a[i][j]+a[i][j-1],a[i][j]+a[i-1][j]),a[i][j]+a[i-1][j-1]);

cout<<a[n][m]<<endl;

return 0;

}

|  |
| --- |
| #include<fstream> #include<vector> #include<queue> using namespace std; ifstream in("winter.in"); ofstream out("winter.out"); bool used[100001]; vector<vector<int> > a; int n,m,i,x,y; void dfs(int v) {         used[v]=1;         for (vector<int>::iterator q=a[v].begin();q!=a[v].end();++q)                 if (!used[\*q]) dfs(\*q); } int main() {         in>>n>>m;         a.resize(n);         for (i=1;i<=m;i++)         {                 in>>x>>y;                 a[x-1].push\_back(y-1);                 a[y-1].push\_back(x-1);         }         int ans=0;         for (i=0;i<n;i++)         {                 if (used[i]==0)                 {                         ans++;                         dfs(i);                 }         }         out<<ans<<endl; } |

#include<fstream>  
#include<set>  
using namespace std;  
ifstream in("beautiful.in");  
ofstream out("beautiful.out");  
int n,i;  
long long x,j;  
set<long long > a,b,c;  
int main()  
{  
        in>>n;  
        for (i=1;i<=n;i++)  
        {  
                in>>x;  
                for (j=2;j\*j<=x;j++)  
                {  
                        if (x%j==0)  
                        {  
                                x=x/j;  
                                b=c=a;  
                                b.insert(j);  
                                if (b==c)  
                                {  
                                        out<<j<<endl;  
                                        return 0;  
                                }  
                                a.insert(j);  
                                if (x%j==0)  
                                {  
                                        out<<j<<endl;  
                                       return 0;  
                                }  
                        }  
                }  
                if (x!=1)  
                {  
                j=x;  
                if (x%j==0)  
                        {  
                                b=c=a;  
                                b.insert(j);  
                                if (b==c)  
                                {  
                                        out<<j<<endl;  
                                        return 0;  
                                }  
                                a.insert(j);  
                                if ((x/j)%j==0)  
                                {  
                                        out<<j<<endl;  
                                        return 0;  
  
                                }  
                        }  
                }  
  
        }  
        out<<"Beautiful"<<endl;  
}