236/A

A. Boy or Girl

time limit per test

1 second

memory limit per test

256 megabytes

input

standard input

output

standard output

Those days, many boys use beautiful girls' photos as avatars in forums. So it is pretty hard to tell the gender of a user at the first glance. Last year, our hero went to a forum and had a nice chat with a beauty (he thought so). After that they talked very often and eventually they became a couple in the network.

But yesterday, he came to see "her" in the real world and found out "she" is actually a very strong man! Our hero is very sad and he is too tired to love again now. So he came up with a way to recognize users' genders by their user names.

This is his method: if the number of distinct characters in one's user name is odd, then he is a male, otherwise she is a female. You are given the string that denotes the user name, please help our hero to determine the gender of this user by his method.

**Input**

The first line contains a non-empty string, that contains only lowercase English letters — the user name. This string contains at most 100 letters.

**Output**

If it is a female by our hero's method, print "CHAT WITH HER!" (without the quotes), otherwise, print "IGNORE HIM!" (without the quotes).

**Examples**

**input**

wjmzbmr

**output**

CHAT WITH HER!

**input**

xiaodao

**output**

IGNORE HIM!

**input**

sevenkplus

**output**

CHAT WITH HER!

**Note**

For the first example. There are 6 distinct characters in "wjmzbmr". These characters are: "w", "j", "m", "z", "b", "r". So wjmzbmr is a female and you should print "CHAT WITH HER!".

#include<iostream>

#include<bits/stdc++.h>

using namespace std;

int main()

{

string s;

int c=0;

cin>>s;

sort(s.begin(),s.end());

for(int i=0;s[i]!='\0';i++)

{

if(s[i]!=s[i+1])

c+=1;

}

if((c%2)==0)

cout<<"CHAT "<<"WITH "<<"HER!"<<endl;

else

cout<<"IGNORE "<<"HIM!"<<endl;

return 0;

}

<http://codeforces.com/contest/505/problem/B>

B. Mr. Kitayuta's Colorful Graph

time limit per test

1 second

memory limit per test

256 megabytes

input

standard input

output

standard output

Mr. Kitayuta has just bought an undirected graph consisting of *n* vertices and *m* edges. The vertices of the graph are numbered from 1 to *n*. Each edge, namely edge *i*, has a color *ci*, connecting vertex *ai* and *bi*.

Mr. Kitayuta wants you to process the following *q* queries.

In the *i*-th query, he gives you two integers — *ui* and *vi*.

Find the number of the colors that satisfy the following condition: the edges of that color connect vertex *ui* and vertex *vi* directly or indirectly.

**Input**

The first line of the input contains space-separated two integers — *n* and *m* (2 ≤ *n* ≤ 100, 1 ≤ *m* ≤ 100), denoting the number of the vertices and the number of the edges, respectively.

The next *m* lines contain space-separated three integers — *ai*, *bi* (1 ≤ *ai* < *bi* ≤ *n*) and *ci* (1 ≤ *ci* ≤ *m*). Note that there can be multiple edges between two vertices. However, there are no multiple edges of the same color between two vertices, that is, if *i* ≠ *j*, (*ai*, *bi*, *ci*) ≠ (*aj*, *bj*, *cj*).

The next line contains a integer — *q* (1 ≤ *q* ≤ 100), denoting the number of the queries.

Then follows *q* lines, containing space-separated two integers — *ui* and *vi* (1 ≤ *ui*, *vi* ≤ *n*). It is guaranteed that *ui* ≠ *vi*.

**Output**

For each query, print the answer in a separate line.

**Examples**

**input**

4 5  
1 2 1  
1 2 2  
2 3 1  
2 3 3  
2 4 3  
3  
1 2  
3 4  
1 4

**output**

2  
1  
0

**input**

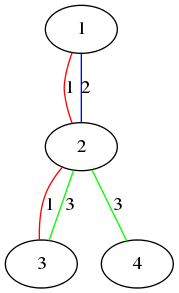
5 7  
1 5 1  
2 5 1  
3 5 1  
4 5 1  
1 2 2  
2 3 2  
3 4 2  
5  
1 5  
5 1  
2 5  
1 5  
1 4

**output**

1  
1  
1  
1  
2

**Note**

Let's consider the first sample.

The figure above shows the first sample.

* Vertex 1 and vertex 2 are connected by color 1 and 2.
* Vertex 3 and vertex 4 are connected by color 3.
* Vertex 1 and vertex 4 are not connected by any single color.

#include <cstdio>

#include <cstring>

using namespace std;

#define N 105

int head[N][N], idx;struct Edge{

int to, next;}edges[N \* 2];bool vis[N];void add(int from, int to, int color){

++idx;

edges[idx].to = to, edges[idx].next = head[from][color], head[from][color] = idx;}bool is\_connected(int u, int v, int color){

vis[u] = true;

for(int i = head[u][color]; i != 0; i = edges[i].next){

int to = edges[i].to;

if(to == v)

return true;

if(!vis[to] && is\_connected(to, v, color))

return true

;

}

return false;}int main(){

int n, m, q;

scanf("%d %d", &n, &m);

for(int i = 0; i < m; ++i){

int v1, v2, color;

scanf("%d %d %d", &v1, &v2, &color);

add(v1, v2, color);

add(v2, v1, color);

}

scanf("%d", &q);

for(int i = 0; i < q; ++i){

int u, v, ans = 0;

scanf("%d %d", &u, &v);

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

memset(vis, false, sizeof(vis));

if(is\_connected(u, v, j))

ans ++;

}

printf("%d\n", ans);

}

return 0;}

<http://codeforces.com/contest/18/problem/A>

A. Triangle

time limit per test

2 seconds

memory limit per test

64 megabytes

input

standard input

output

standard output

At a geometry lesson Bob learnt that a triangle is called right-angled if it is nondegenerate and one of its angles is right. Bob decided to draw such a triangle immediately: on a sheet of paper he drew three points with integer coordinates, and joined them with segments of straight lines, then he showed the triangle to Peter. Peter said that Bob's triangle is not right-angled, but is *almost* right-angled: the triangle itself is not right-angled, but it is possible to move one of the points exactly by distance 1 so, that all the coordinates remain integer, and the triangle become right-angled. Bob asks you to help him and find out if Peter tricks him. By the given coordinates of the triangle you should find out if it is right-angled, almost right-angled, or neither of these.

**Input**

The first input line contains 6 space-separated integers *x*1, *y*1, *x*2, *y*2, *x*3, *y*3 — coordinates of the triangle's vertices. All the coordinates are integer and don't exceed 100 in absolute value. It's guaranteed that the triangle is nondegenerate, i.e. its total area is not zero.

**Output**

If the given triangle is right-angled, output RIGHT, if it is almost right-angled, output ALMOST, and if it is neither of these, output NEITHER.

**Examples**

**input**

0 0 2 0 0 1

**output**

RIGHT

**input**

2 3 4 5 6 6

**output**

NEITHER

**input**

-1 0 2 0 0 1

**output**

ALMOST

#include<stdio.h>

int ans(int x1,int y1,int x2,int y2,int x3,int y3);

int main()

{

int a1,a2,b1,b2,c1,c2;

while(scanf("%d %d %d %d %d %d",&a1,&a2,&b1,&b2,&c1,&c2)!=EOF)

{

if(ans(a1,a2,b1,b2,c1,c2)==1)printf("RIGHT\n");

else

{

int ta,tb,flag=0;

ta=a1+0;

tb=a2+1;

if(ans(ta,tb,b1,b2,c1,c2)==1)flag=1;

ta=a1+1;

tb=a2+0;

if(ans(ta,tb,b1,b2,c1,c2)==1)flag=1;

ta=a1-1;

tb=a2+0;

if(ans(ta,tb,b1,b2,c1,c2)==1)flag=1;

ta=a1+0;

tb=a2-1;

if(ans(ta,tb,b1,b2,c1,c2)==1)flag=1;

ta=b1+1;

tb=b2+0;

if(ans(ta,tb,a1,a2,c1,c2)==1)flag=1;

ta=b1-1;

tb=b2+0;

if(ans(ta,tb,a1,a2,c1,c2)==1)flag=1;

ta=b1+0;

tb=b2+1;

if(ans(ta,tb,a1,a2,c1,c2)==1)flag=1;

ta=b1+0;

tb=b2-1;

if(ans(ta,tb,a1,a2,c1,c2)==1)flag=1;

ta=c1+1;

tb=c2+0;

if(ans(ta,tb,a1,a2,b1,b2)==1)flag=1;

ta=c1+0;

tb=c2+1;

if(ans(ta,tb,a1,a2,b1,b2)==1)flag=1;

ta=c1-1;

tb=c2+0;

if(ans(ta,tb,a1,a2,b1,b2)==1)flag=1;

ta=c1+0;

tb=c2-1;

if(ans(ta,tb,a1,a2,b1,b2)==1)flag=1;

if(flag==1)printf("ALMOST\n");

else printf("NEITHER\n");

}

}

}

int ans(int x1,int y1,int x2,int y2,int x3,int y3)

{

int a,b,c;

a=(x1-x2)\*(x1-x2)+(y1-y2)\*(y1-y2);

b=(x1-x3)\*(x1-x3)+(y1-y3)\*(y1-y3);

c=(x3-x2)\*(x3-x2)+(y3-y2)\*(y3-y2);

if(a==0||b==0||c==0) return 0;

if(a+b==c||a+c==b||b+c==a) return 1;

else return 0;

}

<http://codeforces.com/contest/610/problem/A>

A. Pasha and Stick

time limit per test

1 second

memory limit per test

256 megabytes

input

standard input

output

standard output

Pasha has a wooden stick of some positive integer length *n*. He wants to perform exactly three cuts to get four parts of the stick. Each part must have some positive integer length and the sum of these lengths will obviously be *n*.

Pasha likes rectangles but hates squares, so he wonders, how many ways are there to split a stick into four parts so that it's possible to form a rectangle using these parts, but is impossible to form a square.

Your task is to help Pasha and count the number of such ways. Two ways to cut the stick are considered distinct if there exists some integer *x*, such that the number of parts of length *x* in the first way differ from the number of parts of length *x* in the second way.

**Input**

The first line of the input contains a positive integer *n* (1 ≤ *n* ≤ 2·109) — the length of Pasha's stick.

**Output**

The output should contain a single integer — the number of ways to split Pasha's stick into four parts of positive integer length so that it's possible to make a rectangle by connecting the ends of these parts, but is impossible to form a square.

**Examples**

**input**

6

**output**

1

**input**

20

**output**

4

**Note**

There is only one way to divide the stick in the first sample {1, 1, 2, 2}.

Four ways to divide the stick in the second sample are {1, 1, 9, 9}, {2, 2, 8, 8}, {3, 3, 7, 7} and {4, 4, 6, 6}. Note that {5, 5, 5, 5} doesn't work.

#include <iostream>

#include<string.h>

#include<algorithm>

#include<cmath>

#include<string>

using namespace std;

int main()

{

long long int m=0,n=0,l=1,y,z,x=0,s=0,i,j=0,k,a[110];

cin>>n;

if(n%2==0)

cout<<(n-2)/4;

else

cout<<0;

// your code goes here

return 0;

}

<http://codeforces.com/contest/604/problem/A>

A. Uncowed Forces

time limit per test

1 second

memory limit per test

256 megabytes

input

standard input

output

standard output

Kevin Sun has just finished competing in Codeforces Round #334! The round was 120 minutes long and featured five problems with maximum point values of 500, 1000, 1500, 2000, and 2500, respectively. Despite the challenging tasks, Kevin was uncowed and bulldozed through all of them, distinguishing himself from the herd as the best cowmputer scientist in all of Bovinia. Kevin knows his submission time for each problem, the number of wrong submissions that he made on each problem, and his total numbers of successful and unsuccessful hacks. Because Codeforces scoring is complicated, Kevin wants you to write a program to compute his final score.

Codeforces scores are computed as follows: If the maximum point value of a problem is *x*, and Kevin submitted correctly at minute *m*but made *w* wrong submissions, then his score on that problem is http://codeforces.com/predownloaded/d6/7f/d67fcf29c2911addc12151730c2edc148d49e425.png. His total score is equal to the sum of his scores for each problem. In addition, Kevin's total score gets increased by 100 points for each successful hack, but gets decreased by 50 points for each unsuccessful hack.

All arithmetic operations are performed with absolute precision and no rounding. It is guaranteed that Kevin's final score is an integer.

**Input**

The first line of the input contains five space-separated integers *m*1, *m*2, *m*3, *m*4, *m*5, where *mi* (0 ≤ *mi* ≤ 119) is the time of Kevin's last submission for problem *i*. His last submission is always correct and gets accepted.

The second line contains five space-separated integers *w*1, *w*2, *w*3, *w*4, *w*5, where *wi* (0 ≤ *wi* ≤ 10) is Kevin's number of wrong submissions on problem *i*.

The last line contains two space-separated integers *hs* and *hu* (0 ≤ *hs*, *hu* ≤ 20), denoting the Kevin's numbers of successful and unsuccessful hacks, respectively.

**Output**

Print a single integer, the value of Kevin's final score.

**Examples**

**input**

20 40 60 80 100  
0 1 2 3 4  
1 0

**output**

4900

**input**

119 119 119 119 119  
0 0 0 0 0  
10 0

**output**

4930

**Note**

In the second sample, Kevin takes 119 minutes on all of the problems. Therefore, he gets http://codeforces.com/predownloaded/0e/49/0e49cb6d40f25167bc56beffa082e8e4a70e0b4e.png of the points on each problem. So his score from solving problems is http://codeforces.com/predownloaded/68/eb/68eb1da869a3d1ab583bcc170730a2e00e4700ba.png. Adding in 10·100 = 1000points from hacks, his total score becomes 3930 + 1000 = 4930.

#include<bits/stdc++.h>

#include <algorithm>

#include<vector>

#include<list>

#include<utility>

#include<map>

#include<set>

#include<stack>

#include<queue>

#define ll long long int

#define fill(a,v) memset(a,v,sizeof a)

#define FORN(i, n) for(i = 0; i < n; i++)

#define mp make\_pair

#define pb push\_back

#define s(a) sort(a.begin(),a.end())

#define sa(a,n) sort(a,a+(n))

#define ff first

#define ss second

#define elif else if

using namespace std;

ll i,j,n;

int main()

{

ll m[5];

for(i=0;i<5;i++)

cin>>m[i];

ll x;

ll sum=0;

ll pt[5]={500,1000,1500,2000,2500};

for(i=0;i<5;i++)

{

cin>>x;

sum+=max((3\*pt[i])/10,((250-m[i])\*(pt[i]/250)-50\*x));

}

ll hs,hu;

cin>>hs>>hu;

ll ans=sum+hs\*100-hu\*50;

cout<<ans;

return 0;

}

<http://codeforces.com/contest/743/problem/C>

C. Vladik and fractions

time limit per test

1 second

memory limit per test

256 megabytes

input

standard input

output

standard output

Vladik and Chloe decided to determine who of them is better at math. Vladik claimed that for any positive integer *n* he can represent fraction http://codeforces.com/predownloaded/7e/a3/7ea334dc5ba22c666c8f87f28a3dbd96119e1bbd.png as a sum of three distinct positive fractions in form http://codeforces.com/predownloaded/0e/20/0e203d54bfbb1cc777cf935c83d40b57e96ec58d.png.

Help Vladik with that, i.e for a given *n* find three distinct positive integers *x*, *y* and *z* such that http://codeforces.com/predownloaded/43/6a/436aaafa23e0ed17d16e5e10aab481e508f74c7c.png. Because Chloe can't check Vladik's answer if the numbers are large, he asks you to print numbers not exceeding 109.

If there is no such answer, print -1.

**Input**

The single line contains single integer *n* (1 ≤ *n* ≤ 104).

**Output**

If the answer exists, print 3 distinct numbers *x*, *y* and *z* (1 ≤ *x*, *y*, *z* ≤ 109, *x* ≠ *y*, *x* ≠ *z*, *y* ≠ *z*). Otherwise print -1.

If there are multiple answers, print any of them.

**Examples**

**input**

3

**output**

2 7 42

**input**

7

**output**

7 8 56

#include <bits/stdc++.h>

using namespace std;

#define Size(x) ((int)(x).size())

#define pb push\_back

typedef long long ll;

typedef long double ld;

typedef pair<int,int>pii;

const int INF = 1e9 + 10;

int main()

{

ios\_base :: sync\_with\_stdio(false) ,cin.tie(0) , cout.tie(0);

int n;

cin >> n;

if(n == 1) cout << -1 << endl;

else cout << n << ' ' << n+1 << ' ' << n\*(n+1) << endl;

return 0;

}

<http://codeforces.com/contest/437/problem/B>

B. The Child and Set

time limit per test

1 second

memory limit per test

256 megabytes

input

standard input

output

standard output

At the children's day, the child came to Picks's house, and messed his house up. Picks was angry at him. A lot of important things were lost, in particular the favorite set of Picks.

Fortunately, Picks remembers something about his set *S*:

* its elements were distinct integers from 1 to *limit*;
* the value of http://codeforces.com/predownloaded/00/07/00073cd13459c92696eb1000bbae20e3b5bde162.png was equal to *sum*; here *lowbit*(*x*) equals 2*k* where *k* is the position of the first one in the binary representation of *x*. For example, *lowbit*(100102) = 102, *lowbit*(100012) = 12, *lowbit*(100002) = 100002 (binary representation).

Can you help Picks and find any set *S*, that satisfies all the above conditions?

**Input**

The first line contains two integers: *sum*, *limit* (1 ≤ *sum*, *limit* ≤ 105).

**Output**

In the first line print an integer *n* (1 ≤ *n* ≤ 105), denoting the size of *S*. Then print the elements of set *S* in any order. If there are multiple answers, print any of them.

If it's impossible to find a suitable set, print -1.

**Examples**

**input**

5 5

**output**

2  
4 5

**input**

4 3

**output**

3  
2 3 1

**input**

5 1

**output**

-1

**Note**

In sample test 1: *lowbit*(4) = 4, *lowbit*(5) = 1, 4 + 1 = 5.

In sample test 2: *lowbit*(1) = 1, *lowbit*(2) = 2, *lowbit*(3) = 1, 1 + 2 + 1 = 4.

#include <bits/stdc++.h>

using namespace std;

int main()

{

int sum = 0;

int limit = 0;

vector<int> ans;

vector< pair<int, int> > s;

cin >> sum >> limit;

for (int i = limit; i > 0; --i) {

s.push\_back({i & -i, i});

}

sort(s.begin(), s.end());

for (int i = limit - 1; i >= 0; --i) {

if (s[i].first <= sum) {

sum -= s[i].first;

ans.push\_back(s[i].second);

}

}

if (sum) cout << -1 << '\n';

else {

int size = int(ans.size());

cout << size << '\n';

for (int i = 0; i < size; ++i) cout << ans[i] << ' ';

cout << '\n';

}

return 0;

}

<http://codeforces.com/contest/268/problem/B>

B. Buttons

time limit per test

1 second

memory limit per test

256 megabytes

input

standard input

output

standard output

Manao is trying to open a rather challenging lock. The lock has *n* buttons on it and to open it, you should press the buttons in a certain order to open the lock. When you push some button, it either stays pressed into the lock (that means that you've guessed correctly and pushed the button that goes next in the sequence), or all pressed buttons return to the initial position. When all buttons are pressed into the lock at once, the lock opens.

Consider an example with three buttons. Let's say that the opening sequence is: {2, 3, 1}. If you first press buttons 1 or 3, the buttons unpress immediately. If you first press button 2, it stays pressed. If you press 1 after 2, all buttons unpress. If you press 3 after 2, buttons 3 and 2 stay pressed. As soon as you've got two pressed buttons, you only need to press button 1 to open the lock.

Manao doesn't know the opening sequence. But he is really smart and he is going to act in the optimal way. Calculate the number of times he's got to push a button in order to open the lock in the worst-case scenario.

**Input**

A single line contains integer *n* (1 ≤ *n* ≤ 2000) — the number of buttons the lock has.

**Output**

In a single line print the number of times Manao has to push a button in the worst-case scenario.

**Examples**

**input**

2

**output**

3

**input**

3

**output**

7

**Note**

Consider the first test sample. Manao can fail his first push and push the wrong button. In this case he will already be able to guess the right one with his second push. And his third push will push the second right button. Thus, in the worst-case scenario he will only need 3 pushes.

#include <iostream>

using namespace std;

int main(void)

{

int n;

cin>>n;

int sum= n\*(n+1)/2;

// n-= 2;

// if(n>0) sum+= n\*(n+1)/2;

int j;

for(int i= 2; i<n; i++){

j= n-i;

sum+= j\*(j+1)/2;

}

cout<< sum <<endl;

return 0;

}

<http://codeforces.com/contest/255/problem/A>

A. Greg's Workout

time limit per test

2 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

Greg is a beginner bodybuilder. Today the gym coach gave him the training plan. All it had was *n* integers *a*1, *a*2, ..., *an*. These numbers mean that Greg needs to do exactly *n* exercises today. Besides, Greg should repeat the *i*-th in order exercise *ai* times.

Greg now only does three types of exercises: "chest" exercises, "biceps" exercises and "back" exercises. Besides, his training is cyclic, that is, the first exercise he does is a "chest" one, the second one is "biceps", the third one is "back", the fourth one is "chest", the fifth one is "biceps", and so on to the *n*-th exercise.

Now Greg wonders, which muscle will get the most exercise during his training. We know that the exercise Greg repeats the maximum number of times, trains the corresponding muscle the most. Help Greg, determine which muscle will get the most training.

**Input**

The first line contains integer *n* (1 ≤ *n* ≤ 20). The second line contains *n* integers *a*1, *a*2, ..., *an* (1 ≤ *ai* ≤ 25) — the number of times Greg repeats the exercises.

**Output**

Print word "chest" (without the quotes), if the chest gets the most exercise, "biceps" (without the quotes), if the biceps gets the most exercise and print "back" (without the quotes) if the back gets the most exercise.

It is guaranteed that the input is such that the answer to the problem is **unambiguous**.

**Examples**

**input**

2  
2 8

**output**

biceps

**input**

3  
5 1 10

**output**

back

**input**

7  
3 3 2 7 9 6 8

**output**

chest

**Note**

In the first sample Greg does 2 chest, 8 biceps and zero back exercises, so the biceps gets the most exercises.

In the second sample Greg does 5 chest, 1 biceps and 10 back exercises, so the back gets the most exercises.

In the third sample Greg does 18 chest, 12 biceps and 8 back exercises, so the chest gets the most exercise.

#include <cstdio>

#include <algorithm>

const int N = 30;

int arr[N], n, c1, c2, c3, max;

int main()

{

scanf("%d", &n);

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

{

scanf("%d", &arr[i]);

switch (i % 3)

{

case 1:

c1 += arr[i];

break;

case 2:

c2 += arr[i];

break;

case 0:

c3 += arr[i];

break;

}

}

if (std::max(c1, std::max(c2, c3)) == c1)

{

printf("chest"); return 0;

}

else if (std::max(c1, std::max(c2, c3)) == c2)

{

printf("biceps"); return 0;

}

else if (std::max(c1, std::max(c2, c3)) == c3)

{

printf("back"); return 0;

}

}

<http://codeforces.com/contest/219/problem/A>

A. k-String

time limit per test

2 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

A string is called a *k*-string if it can be represented as *k* concatenated copies of some string. For example, the string "aabaabaabaab" is at the same time a 1-string, a 2-string and a 4-string, but it is not a 3-string, a 5-string, or a 6-string and so on. Obviously any string is a 1-string.

You are given a string *s*, consisting of lowercase English letters and a positive integer *k*. Your task is to reorder the letters in the string *s*in such a way that the resulting string is a *k*-string.

**Input**

The first input line contains integer *k* (1 ≤ *k* ≤ 1000). The second line contains *s*, all characters in *s* are lowercase English letters. The string length *s* satisfies the inequality 1 ≤ |*s*| ≤ 1000, where |*s*| is the length of string *s*.

**Output**

Rearrange the letters in string *s* in such a way that the result is a *k*-string. Print the result on a single output line. If there are multiple solutions, print any of them.

If the solution doesn't exist, print "-1" (without quotes).

**Examples**

**input**

2  
aazz

**output**

azaz

**input**

3  
abcabcabz

**output**

-1

#include <bits/stdc++.h>

#define eps 1e-8

#define inf 0x3f3f3f3f

#define INF 2e18

#define LL long long

#define ULL unsigned long long

#define PI acos(-1.0)

#define pb push\_back

#define mk make\_pair

#define pii pair<int,int>

#define pLL pair<LL,LL>

#define ff first

#define ss second

#define all(a) a.begin(),a.end()

#define SQR(a) ((a)\*(a))

#define Unique(a) sort(all(a)),a.erase(unique(all(a)),a.end())

#define min3(a,b,c) min(a,min(b,c))

#define max3(a,b,c) max(a,max(b,c))

#define min4(a,b,c,d) min(min(a,b),min(c,d))

#define max4(a,b,c,d) max(max(a,b),max(c,d))

#define print freopen("out.txt","w",stdout);

int left(int n)

{

return n<<1;

}

int right(int n)

{

return (n<<1)+1;

}

namespace patch

{

template<typename T>std::string to\_string(const T&n)

{

std::ostringstream stm;

stm<<n;

return stm.str();

}

}

using namespace std;

int Set(int N,int pos)

{

return N=N | (1<<pos);

}

int reset(int N,int pos)

{

return N= N & ~(1<<pos);

}

bool check(int N,int pos)

{

return (bool)(N & (1<<pos));

}

int ii[]= {-2,-2,-3,-1,-1,+1};

int jj[]= {+1,-1,-1,-2,-3,-2};

template <typename T> inline T GCD (T a,T b )

{

a = abs(a);

b = abs(b);

while ( b )

{

a = a % b;

swap ( a, b );

}

return a;

}

template <typename T> inline T LCM(T x,T y)

{

T tp = GCD(x,y);

if( (x / tp) \* 1. \* y > 9e18) return 9e18;

return (x / tp) \* y;

}

template <typename T> inline T BigMod(T A,T B,T M)

{

T ret = 1;

while(B)

{

if(B & 1) ret = (ret \* A) % M;

A = (A \* A) % M;

B = B >> 1;

}

return ret;

}

template <typename T> inline T ModInv (T A,T M)

{

return BigMod(A,M-2,M);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* End of template \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int a[100010],b[100010],c[100010];

int main()

{

int k,mx=0;

cin>>k;

string str;

cin>>str;

int len=str.size();

for(int i=0;i<str.size();i++)

{

a[str[i]-'a']++;

b[str[i]-'a']++;

}

for(int i=0;i<26;i++)

{

if(a[i]!=0){}

{

if(a[i]%k!=0) return cout<<-1,0;

}

}

while(len>0)

{

for(int i=0;i<26;i++)

{

if(b[i]>0){

b[i]-=a[i]/k;

for(int j=0;j<a[i]/k;j++) printf("%c",i+'a'),len--;

}

}

}

}

<http://codeforces.com/contest/41/problem/A>

A. Translation

time limit per test

2 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

The translation from the Berland language into the Birland language is not an easy task. Those languages are very similar: a berlandish word differs from a birlandish word with the same meaning a little: it is spelled (and pronounced) reversely. For example, a Berlandish word code corresponds to a Birlandish word edoc. However, it's easy to make a mistake during the «translation». Vasya translated word *s* from Berlandish into Birlandish as *t*. Help him: find out if he translated the word correctly.

**Input**

The first line contains word *s*, the second line contains word *t*. The words consist of lowercase Latin letters. The input data do not consist unnecessary spaces. The words are not empty and their lengths do not exceed 100 symbols.

**Output**

If the word *t* is a word *s*, written reversely, print YES, otherwise print NO.

**Examples**

**input**

code  
edoc

**output**

YES

**input**

abb  
aba

**output**

NO

**input**

code  
code

**output**

NO

#include <iostream>

using namespace std;

int main()

{

string s, t;

int a, b, c=0;

cin>>s;

cin>>t;

for(a=1; a<1000; a++)

{

if(s[a]=='\0')

break;

}

for(b=0; b<a; b++)

{

if(s[b]!=t[a-b-1])

{

c=1;

break;

}

else

continue;

}

if(c==1)

cout<<"NO";

else if(c==0)

cout<<"YES";

return 0;

}

<http://codeforces.com/contest/752/problem/B>

B. Santa Claus and Keyboard Check

time limit per test

2 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

Santa Claus decided to disassemble his keyboard to clean it. After he returned all the keys back, he suddenly realized that some pairs of keys took each other's place! That is, Santa suspects that each key is either on its place, or on the place of another key, which is located exactly where the first key should be.

In order to make sure that he's right and restore the correct order of keys, Santa typed his favorite patter looking only to his keyboard.

You are given the Santa's favorite patter and the string he actually typed. Determine which pairs of keys could be mixed. Each key must occur in pairs **at most once**.

**Input**

The input consists of only two strings *s* and *t* denoting the favorite Santa's patter and the resulting string. *s* and *t* are not empty and have the same length, which is at most 1000. Both strings consist only of lowercase English letters.

**Output**

If Santa is wrong, and there is no way to divide some of keys into pairs and swap keys in each pair so that the keyboard will be fixed, print «-1» (without quotes).

Otherwise, the first line of output should contain the only integer *k* (*k* ≥ 0) — the number of pairs of keys that should be swapped. The following *k* lines should contain two space-separated letters each, denoting the keys which should be swapped. All printed letters must be distinct.

If there are several possible answers, print any of them. You are free to choose the order of the pairs and the order of keys in a pair.

Each letter must occur at most once. Santa considers the keyboard to be fixed if he can print his favorite patter without mistakes.

**Examples**

**input**

helloworld  
ehoolwlroz

**output**

3  
h e  
l o  
d z

**input**

hastalavistababy  
hastalavistababy

**output**

0

**input**

merrychristmas  
christmasmerry

**output**

-1

#include <bits/stdc++.h>

using namespace std;

int main(int argc, char \*argv[]){

//freopen("inp.txt", "r", stdin);

string from, to;

set<char> trues;

map<char, char> changes, res;

while(cin >> from >> to){

changes.clear();

res.clear();

trues.clear();

bool r = true;

for(int i = 0; i < from.length(); i++)

if(from[i] == to[i])

trues.insert(from[i]);

for(int i = 0; i < from.length() && r; i++)

if(from[i] != to[i]){

if(trues.find(from[i]) != trues.end()

|| trues.find(to[i]) != trues.end()){

r = false;

break;

}

else if(changes.find(from[i]) != changes.end()){

if(changes.find(to[i]) != changes.end()){

if(changes[from[i]] != to[i]){

r = false;

break;

}

}

else{

r = false;

break;

}

}

else if(changes.find(to[i]) != changes.end()){

if(changes[to[i]] != from[i]){

r = false;

break;

}

}

else{

if(changes.find(to[i]) == changes.end()){

changes[from[i]] = to[i];

changes[to[i]] = from[i];

}

}

}

if(!r){

cout << "-1" << endl;

continue;

}

for(auto v : changes){

if(res.find(v.first) != res.end() || res.find(v.second) != res.end() )

continue;

res[v.first] = v.second;

}

cout << res.size() << endl;

for(auto v : res){

cout << v.first << " " << v.second << endl;

}

}

return 0;

}

<http://codeforces.com/contest/749/problem/A>

A. Bachgold Problem

time limit per test

1 second

memory limit per test

256 megabytes

input

standard input

output

standard output

Bachgold problem is very easy to formulate. Given a positive integer *n* represent it as a sum of **maximum possible** number of prime numbers. One can prove that such representation exists for any integer greater than 1.

Recall that integer *k* is called *prime* if it is greater than 1 and has exactly two positive integer divisors — 1 and *k*.

**Input**

The only line of the input contains a single integer *n* (2 ≤ *n* ≤ 100 000).

**Output**

The first line of the output contains a single integer *k* — maximum possible number of primes in representation.

The second line should contain *k* primes with their sum equal to *n*. You can print them in any order. If there are several optimal solution, print any of them.

**Examples**

**input**

5

**output**

2  
2 3

**input**

6

**output**

3  
2 2 2

#include<bits/stdc++.h>

using namespace std;

int main()

{

int n;

cin>>n;

if(n%2==0)

{

cout<<n/2<<endl;

for(int i=1; i<=n/2; i++)

{

cout<<2<<" ";

}

}

else

{

cout<<n/2<<endl;

for(int i=1; i<=n/2-1; i++)

{

cout<<2<<" ";

}

cout<<3;

}

return 0;

}

<http://codeforces.com/contest/722/problem/B>

B. Verse Pattern

time limit per test

1 second

memory limit per test

256 megabytes

input

standard input

output

standard output

You are given a text consisting of *n* lines. Each line contains some space-separated words, consisting of lowercase English letters.

We define a syllable as a string that contains exactly one vowel and any arbitrary number (possibly none) of consonants. In English alphabet following letters are considered to be vowels: 'a', 'e', 'i', 'o', 'u' and 'y'.

Each word of the text that contains at least one vowel can be divided into syllables. Each character should be a part of exactly one syllable. For example, the word "mamma" can be divided into syllables as "ma" and "mma", "mam" and "ma", and "mamm" and "a". Words that consist of only consonants should be ignored.

The verse patterns for the given text is a sequence of *n* integers *p*1, *p*2, ..., *pn*. Text matches the given verse pattern if for each *i* from 1to *n* one can divide words of the *i*-th line in syllables in such a way that the total number of syllables is equal to *pi*.

You are given the text and the verse pattern. Check, if the given text matches the given verse pattern.

**Input**

The first line of the input contains a single integer *n* (1 ≤ *n* ≤ 100) — the number of lines in the text.

The second line contains integers *p*1, ..., *pn* (0 ≤ *pi* ≤ 100) — the verse pattern.

Next *n* lines contain the text itself. Text consists of lowercase English letters and spaces. It's guaranteed that all lines are non-empty, each line starts and ends with a letter and words are separated by exactly one space. The length of each line doesn't exceed 100characters.

**Output**

If the given text matches the given verse pattern, then print "YES" (without quotes) in the only line of the output. Otherwise, print "NO" (without quotes).

**Examples**

**input**

3  
2 2 3  
intel  
code  
ch allenge

**output**

YES

**input**

4  
1 2 3 1  
a  
bcdefghi  
jklmnopqrstu  
vwxyz

**output**

NO

**input**

4  
13 11 15 15  
to be or not to be that is the question  
whether tis nobler in the mind to suffer  
the slings and arrows of outrageous fortune  
or to take arms against a sea of troubles

**output**

YES

**Note**

In the first sample, one can split words into syllables in the following way:

in-tel  
co-de  
ch al-len-ge

Since the word "ch" in the third line doesn't contain vowels, we can ignore it. As the result we get 2 syllabels in first two lines and 3syllables in the third one.

#include <algorithm>

#include <bitset>

#include <cassert>

#include <cctype>

#include <climits>

#include <cmath>

#include <cstdio>

#include <cstdlib>

#include <cstring>

#include <ctime>

#include <iomanip>

#include <ios>

#include <iostream>

#include <map>

#include <queue>

#include <set>

#include <unistd.h>

#include <utility>

#include <vector>

#define dbg(args...) //fprintf(stderr, args)

#define dbc(x) cerr << x << "\n"

#define dbn(x) cerr << #x << " == " << x << "\n"

#define m(x) memset(x,0,sizeof(x))

#define m1(x) memset(x,-1,sizeof(x))

#define minf(x) memset(x,63,sizeof(x))

#define pb push\_back

#define eb emplace\_back

#define F first

#define S second

using namespace std;

typedef long long ll;

typedef pair<int,int> pii;

const int INF = 0x3f3f3f3f;

//

vector<int> v;

set<char> vo = {'a', 'e', 'i', 'o', 'u', 'y'};

int main(){

ios::sync\_with\_stdio(false);

int n; cin >> n;

for (int i=0; i<n; i++) {

int x; cin >> x;

v.pb(x);

}

string s;

getline(cin, s);

for (int i=0; i<n; i++) {

getline(cin, s);

int count = 0;

for (int j=0; j<s.size(); j++) {

if (vo.find(s[j]) != vo.end()) {

count++;

}

}

if (count != v[i]) {

cout << "NO" << endl;

return 0;

}

}

cout << "YES" << endl;

return 0;

}

<http://codeforces.com/contest/158/problem/B>

B. Taxi

time limit per test

3 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

After the lessons *n* groups of schoolchildren went outside and decided to visit Polycarpus to celebrate his birthday. We know that the *i*-th group consists of *si* friends (1 ≤ *si* ≤ 4), and they want to go to Polycarpus together. They decided to get there by taxi. Each car can carry at most four passengers. What minimum number of cars will the children need if all members of each group should ride in the same taxi (but one taxi can take more than one group)?

**Input**

The first line contains integer *n* (1 ≤ *n* ≤ 105) — the number of groups of schoolchildren. The second line contains a sequence of integers *s*1, *s*2, ..., *sn* (1 ≤ *si* ≤ 4). The integers are separated by a space, *si* is the number of children in the *i*-th group.

**Output**

Print the single number — the minimum number of taxis necessary to drive all children to Polycarpus.

**Examples**

**input**

5  
1 2 4 3 3

**output**

4

**input**

8  
2 3 4 4 2 1 3 1

**output**

5

**Note**

In the first test we can sort the children into four cars like this:

* the third group (consisting of four children),
* the fourth group (consisting of three children),
* the fifth group (consisting of three children),
* the first and the second group (consisting of one and two children, correspondingly).

There are other ways to sort the groups into four cars.

#include<bits/stdc++.h>

using namespace std;

#define fast ios\_base::sync\_with\_stdio(false)

#define bfast cin.tie(0)

#define outs(x) cout << x << " "

#define outn(x) cout << x << "\n"

#define sf scanf

#define pf printf

#define nl puts("")

#define psb push\_back

typedef long long LL;

typedef vector<int>vii;

typedef vector<LL>vll;

const int mod = 1000007;

const int high = 100003;

int ar[high];

int main()

{

fast;

int n , x , i , j , cnt=0;

while(cin >> n)

{

for(i=0; i<n; i++)

{

cin >> ar[i];

}

sort(ar , ar+n);

i=0;

j=n-1;

cnt=0;

for(i=0, j=n-1; i<=j;)

{

if(i == j)

{

cnt++;

break;

}

int def = 4 - ar[j];

if(ar[i] <= def)

{

i++;

ar[j]+=ar[i];

}

else

{

cnt++;

j--;

}

}

outn(cnt);

}

return 0;

}

<http://codeforces.com/contest/313/problem/B>

B. Ilya and Queries

time limit per test

2 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

Ilya the Lion wants to help all his friends with passing exams. They need to solve the following problem to pass the IT exam.

You've got string *s* = *s*1*s*2... *sn* (*n* is the length of the string), consisting only of characters "." and "#" and *m* queries. Each query is described by a pair of integers *li*, *ri* (1 ≤ *li* < *ri* ≤ *n*). The answer to the query *li*, *ri* is the number of such integers *i* (*li* ≤ *i* < *ri*), that *si* = *si*+ 1.

Ilya the Lion wants to help his friends but is there anyone to help him? Help Ilya, solve the problem.

**Input**

The first line contains string *s* of length *n* (2 ≤ *n* ≤ 105). It is guaranteed that the given string only consists of characters "." and "#".

The next line contains integer *m* (1 ≤ *m* ≤ 105) — the number of queries. Each of the next *m* lines contains the description of the corresponding query. The *i*-th line contains integers *li*, *ri* (1 ≤ *li* < *ri* ≤ *n*).

**Output**

Print *m* integers — the answers to the queries in the order in which they are given in the input.

**Examples**

**input**

......  
4  
3 4  
2 3  
1 6  
2 6

**output**

1  
1  
5  
4

**input**

#..###  
5  
1 3  
5 6  
1 5  
3 6  
3 4

**output**

1  
1  
2  
2  
0

#include<iostream>

using namespace std;

int main()

{

string s;

cin>>s;

int n = s.length(),i;

bool arr[n] = {0};

for(i=0;i<n-1;i++)

{

if(s[i]==s[i+1])

arr[i]=1;

}

int m,l,r,ans,j;

cin>>m;

int sum[n];

int cur\_sum=0;

for(i=0;i<n;i++)

{

cur\_sum += arr[i];

sum[i]=cur\_sum;

}

// for(i=0;i<n;i++)

// cout<<sum[i];

for(i=0;i<m;i++)

{

ans=0;

cin>>l>>r;

if(l==1)

cout<<sum[r-2]<<endl;

else

cout<<(sum[r-2]-sum[l-2])<<endl;

}

}

<http://codeforces.com/contest/752/problem/C>

C. Santa Claus and Robot

time limit per test

2 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

Santa Claus has Robot which lives on the infinite grid and can move **along its lines**. He can also, having a sequence of *m* points *p*1, *p*2, ..., *pm* with integer coordinates, do the following: denote its initial location by *p*0. First, the robot will move from *p*0 to *p*1 along one of the shortest paths between them (please notice that since the robot moves only along the grid lines, there can be several shortest paths). Then, after it reaches *p*1, it'll move to *p*2, again, choosing one of the shortest ways, then to *p*3, and so on, until he has visited all points in the given order. Some of the points in the sequence may coincide, in that case Robot will visit that point several times according to the sequence order.

While Santa was away, someone gave a sequence of points to Robot. This sequence is now lost, but Robot saved the protocol of its unit movements. Please, find the minimum possible length of the sequence.

**Input**

The first line of input contains the only positive integer *n* (1 ≤ *n* ≤ 2·105) which equals the number of unit segments the robot traveled. The second line contains the movements protocol, which consists of *n* letters, each being equal either L, or R, or U, or D. *k*-th letter stands for the direction which Robot traveled the *k*-th unit segment in: L means that it moved to the left, R — to the right, U — to the top and D — to the bottom. Have a look at the illustrations for better explanation.

**Output**

The only line of input should contain the minimum possible length of the sequence.

**Examples**

**input**

4  
RURD

**output**

2

**input**

6  
RRULDD

**output**

2

**input**

26  
RRRULURURUULULLLDLDDRDRDLD

**output**

7

**input**

3  
RLL

**output**

2

**input**

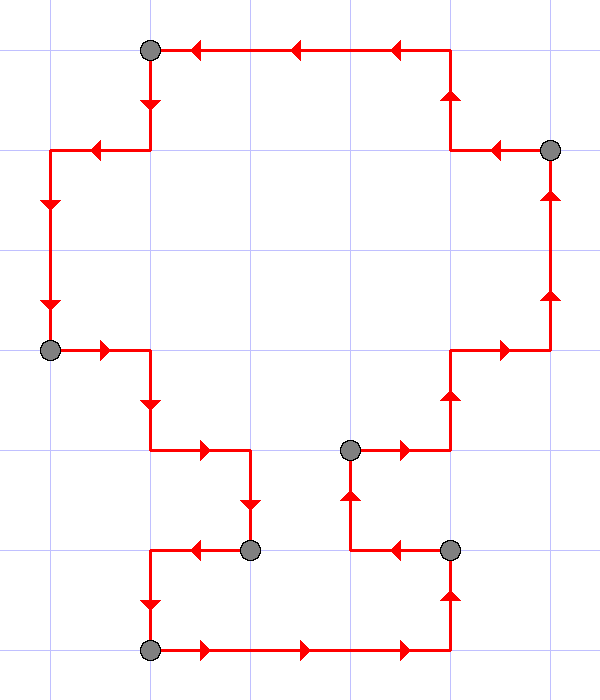
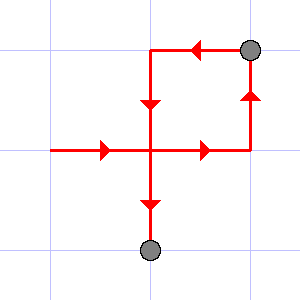
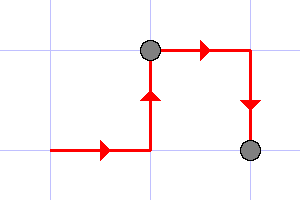
4  
LRLR

**output**

4

**Note**

The illustrations to the first three tests are given below.



The last example illustrates that each point in the sequence should be counted as many times as it is presented in the sequence.

<http://codeforces.com/contest/365/problem/B>

B. The Fibonacci Segment

time limit per test

1 second

memory limit per test

256 megabytes

input

standard input

output

standard output

You have array *a*1, *a*2, ..., *an*. Segment [*l*, *r*] (1 ≤ *l* ≤ *r* ≤ *n*) is good if *ai* = *ai*- 1 + *ai*- 2, for all *i* (*l* + 2 ≤ *i* ≤ *r*).

Let's define *len*([*l*, *r*]) = *r* - *l* + 1, *len*([*l*, *r*]) is the length of the segment [*l*, *r*]. Segment [*l*1, *r*1], is longer than segment [*l*2, *r*2], if *len*([*l*1, *r*1]) > *len*([*l*2, *r*2]).

Your task is to find a good segment of the maximum length in array *a*. Note that a segment of length 1 or 2 is always good.

**Input**

The first line contains a single integer *n* (1 ≤ *n* ≤ 105) — the number of elements in the array. The second line contains integers: *a*1, *a*2, ..., *an* (0 ≤ *ai* ≤ 109).

**Output**

Print the length of the longest good segment in array *a*.

**Examples**

**input**

10  
1 2 3 5 8 13 21 34 55 89

**output**

10

**input**

5  
1 1 1 1 1

**output**

2

#include <algorithm>

#include <assert.h>

#include <bitset>

#include <complex>

#include <ctime>

#include <fstream>

#include <iomanip>

#include <limits.h>

#include <list>

#include <map>

#include <math.h>

#include <queue>

#include <set>

#include <stack>

#include <stdio.h>

#include <string>

#include <unordered\_map>

#include <unordered\_set>

#include <utility>

#include <vector>

using namespace std;

typedef long long LL;

typedef vector<int> VI;

typedef vector<VI> VVI;

typedef vector<LL> VLL;

typedef pair<LL, LL> PLL;

typedef pair<int,int> PII;

typedef pair<int, int> PII;

typedef pair<LL, PLL> PLPLL;

typedef pair<PLL,LL> PPLLL;

typedef pair<LL, char> PLC;

typedef pair<char, char> PCC;

typedef pair<LL, VLL> PVLL;

typedef vector<PLL> VPLL;

typedef vector<PII> VPII;

typedef vector<std::string> VS;

typedef std::vector<LL>::iterator VLLitr;

#define arjun int main()

#define FOR(i, x, y) for (LL i = (x); i < (y); ++i)

#define FORC(i, x, y, in) for (LL i = (x); i < (y); i = i + in)

#define RFOR(i, x, y) for (LL i = (x); i >= (y); i--)

#define FORO(it, c) \

for (\_\_typeof((c).begin()) it = (c).begin(); it != (c).end(); it++)

#define pb(e) push\_back(e)

#define mp make\_pair

#define F first

#define S second

#define All(x) x.begin(), x.end()

#define fast \

ios\_base::sync\_with\_stdio(false); \

cin.tie(NULL);

#define PI 3.14159265

#define RESET(a, b) memset(a, b, sizeof(a))

template <typename T> T maxm(T a, T b) { return (a > b) ? a : b; }

template <typename T> T minm(T a, T b) { return (a < b) ? a : b; }

template <typename T> T power(T e, T n) {

T x = 1, p = e;

while (n) {

if (n & 1)

x = x \* p;

p = p \* p;

n >>= 1;

}

return x;

}

template <typename T> T powerm(T e, T n, T m) {

T x = 1, p = e;

while (n) {

if (n & 1)

x = (x \* p) % m;

p = (p \* p) % m;

n >>= 1;

}

return x;

}

template <typename T> T InverseEuler(T a, T m) {

return (a == 1 ? 1 : power(a, m - 2, m));

}

template <typename T> T lcm(T a, T b) { return (a \* (b / \_\_gcd(a, b))); }

string StringToUpper(string strToConvert) {

std::transform(strToConvert.begin(), strToConvert.end(), strToConvert.begin(),

::toupper);

return strToConvert;

}

string StringToLower(string strToConvert) {

std::transform(strToConvert.begin(), strToConvert.end(), strToConvert.begin(),

::tolower);

return strToConvert;

}

LL MAX = (LL)(3 \* 1e18);

LL MOD = (LL)(1e9 + 7);

#ifdef JUDGE

#include <fstream>

std::ifstream cin("input.txt");

std::ofstream cout("output.txt");

#else

#include <iostream>

using std::cin;

using std::cout;

#endif

arjun {

fast;

LL n;

cin>>n;

VLL v(n,0);

LL sum = 0;

LL hun = 0;

LL twoHun = 0;

FOR(i,0,n){

cin>>v[i];

}

LL ans = minm(n,2LL);

LL count = 0;

FOR(i,2,n){

if(v[i] == (v[i-1]+v[i-2])){

count++;

}

else{

ans = maxm(count+2,ans);

count = 0;

}

}

ans = minm(maxm(count+2,ans),n);

cout<<ans<<endl;

}

<http://codeforces.com/contest/752/problem/E>

E. Santa Claus and Tangerines

time limit per test

2 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

Santa Claus has *n* tangerines, and the *i*-th of them consists of exactly *ai* slices. Santa Claus came to a school which has *k* pupils. Santa decided to treat them with tangerines.

However, there can be too few tangerines to present at least one tangerine to each pupil. So Santa decided to divide tangerines into parts so that no one will be offended. In order to do this, he can divide a tangerine or any existing part into two smaller equal parts. If the number of slices in the part he wants to split is odd, then one of the resulting parts will have one slice more than the other. It's forbidden to divide a part consisting of only one slice.

**Santa Claus wants to present to everyone either a whole tangerine or exactly one part of it** (that also means that everyone must get a positive number of slices). One or several tangerines or their parts may stay with Santa.

Let *bi* be the number of slices the *i*-th pupil has in the end. Let Santa's *joy* be the minimum among all *bi*'s.

Your task is to find the maximum possible *joy* Santa can have after he treats everyone with tangerines (or their parts).

**Input**

The first line contains two positive integers *n* and *k* (1 ≤ *n* ≤ 106, 1 ≤ *k* ≤ 2·109) denoting the number of tangerines and the number of pupils, respectively.

The second line consists of *n* positive integers *a*1, *a*2, ..., *an* (1 ≤ *ai* ≤ 107), where *ai* stands for the number of slices the *i*-th tangerine consists of.

**Output**

If there's no way to present a tangerine or a part of tangerine to everyone, print -1. Otherwise, print the maximum possible *joy* that Santa can have.

**Examples**

**input**

3 2  
5 9 3

**output**

5

**input**

2 4  
12 14

**output**

6

**input**

2 3  
1 1

**output**

-1

**Note**

In the first example Santa should divide the second tangerine into two parts with 5 and 4 slices. After that he can present the part with 5slices to the first pupil and the whole first tangerine (with 5 slices, too) to the second pupil.

In the second example Santa should divide both tangerines, so that he'll be able to present two parts with 6 slices and two parts with 7slices.

In the third example Santa Claus can't present 2 slices to 3 pupils in such a way that everyone will have anything.

#include "bits/stdc++.h"

#define MAXN 1000009

#define INF 1000000007

#define mp(x,y) make\_pair(x,y)

#define all(v) v.begin(),v.end()

#define pb(x) push\_back(x)

#define wr cout<<"----------------"<<endl;

#define ppb() pop\_back()

#define tr(ii,c) for(typeof((c).begin()) ii=(c).begin();ii!=(c).end();ii++)

#define ff first

#define ss second

using namespace std;

typedef long long ll;

typedef pair<int,int> PII;

template<class T>bool umin(T& a,T b){if(a>b){a=b;return 1;}return 0;}

template<class T>bool umax(T& a,T b){if(a<b){a=b;return 1;}return 0;}

const int N=1e7+9;

int dp[N],n,k,arr[MAXN];

int rec(int x,int y){

if(x<y)

return 0;

if(x>=y and x<y\*2)

return 1;

int &ret=dp[x];

if(~ret)

return ret;ret=rec(x/2,y);

if(x&1)

return ret+=rec(x-x/2,y);

return ret=ret\*2;

}

bool ok(int x){

memset(dp,-1,sizeof dp);

int res=0;

for(int i=n;i>=1;i--){

res+=rec(arr[i],x);

if(res>=k)

return 1;

}

return 0;

}

int main(){

#ifndef ONLINE\_JUDGE

freopen("file.in", "r", stdin);

#endif

scanf("%d%d",&n,&k);

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

scanf("%d",arr+i);

sort(arr+1,arr+n+1);

if(!ok(1)){

puts("-1");

return 0;

}

int st=1,en=int(1e7);

while(st+1<en){

int mid=(st+en)>>1;

if(ok(mid))

st=mid;

else

en=mid;

}

if(ok(en))

printf("%d\n",en);

else

printf("%d\n",st);

return 0;

}

<http://codeforces.com/contest/547/problem/B>

B. Mike and Feet

time limit per test

1 second

memory limit per test

256 megabytes

input

standard input

output

standard output

Mike is the president of country What-The-Fatherland. There are *n* bears living in this country besides Mike. All of them are standing in a line and they are numbered from 1 to *n* from left to right. *i*-th bear is exactly *ai* feet high.



A group of bears is a non-empty contiguous segment of the line. The *size* of a group is the number of bears in that group. The *strength* of a group is the minimum height of the bear in that group.

Mike is a curious to know for each *x* such that 1 ≤ *x* ≤ *n* the maximum strength among all groups of size *x*.

**Input**

The first line of input contains integer *n* (1 ≤ *n* ≤ 2 × 105), the number of bears.

The second line contains *n* integers separated by space, *a*1, *a*2, ..., *an* (1 ≤ *ai* ≤ 109), heights of bears.

**Output**

Print *n* integers in one line. For each *x* from 1 to *n*, print the maximum strength among all groups of size *x*.

**Examples**

**input**

10  
1 2 3 4 5 4 3 2 1 6

**output**

6 4 4 3 3 2 2 1 1 1

#include <iostream>

#include <cstdio>

#include <vector>

#include <algorithm>

#include <cstring>

#include <string>

#include <queue>

#include <set>

#include <map>

#include <bitset>

#include <cmath>

#include <stack>

#include <complex>

#include <ctime>

#include <unordered\_map>

#pragma comment (linker, "/STACK:256000000")

using namespace std;

const int maxN = 210000;

int n, a[maxN], l[maxN], r[maxN];

int res[maxN];

int main() {

//freopen("input.txt", "r", stdin);

//freopen("output.txt", "w", stdout);

scanf("%d", &n);

for (int i = 0; i < n; ++i) {

scanf("%d", &a[i]);

}

for (int i = 0; i < n; ++i) {

if (i == 0) {

l[i] = i;

continue;

}

int j = i - 1;

while (j >= 0 && a[j] >= a[i]) {

j = l[j] - 1;

}

l[i] = j + 1;

}

for (int i = n - 1; i >= 0; --i) {

if (i == n - 1) {

r[i] = i;

continue;

}

int j = i + 1;

while (j < n && a[j] >= a[i]) {

j = r[j] + 1;

}

r[i] = j - 1;

}

for (int i = 0; i < n; ++i) {

res[r[i] - l[i] + 1] = max(res[r[i] - l[i] + 1], a[i]);

}

for (int i = n - 1; i >= 0; --i) {

res[i] = max(res[i], res[i + 1]);

}

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

printf("%d ", res[i]);

}

printf("\n");

return 0;

}

<http://codeforces.com/contest/747/problem/A>

A. Display Size

time limit per test

1 second

memory limit per test

256 megabytes

input

standard input

output

standard output

A big company decided to launch a new series of rectangular displays, and decided that the display must have exactly *n* pixels.

Your task is to determine the size of the rectangular display — the number of lines (rows) of pixels *a* and the number of columns of pixels *b*, so that:

* there are exactly *n* pixels on the display;
* the number of rows does not exceed the number of columns, it means *a* ≤ *b*;
* the difference *b* - *a* is as small as possible.

**Input**

The first line contains the positive integer *n* (1 ≤ *n* ≤ 106) — the number of pixels display should have.

**Output**

Print two integers — the number of rows and columns on the display.

**Examples**

**input**

8

**output**

2 4

**input**

64

**output**

8 8

**input**

5

**output**

1 5

**input**

999999

**output**

999 1001

**Note**

In the first example the minimum possible difference equals 2, so on the display should be 2 rows of 4 pixels.

In the second example the minimum possible difference equals 0, so on the display should be 8 rows of 8 pixels.

In the third example the minimum possible difference equals 4, so on the display should be 1 row of 5 pixels.

#include <bits/stdc++.h>

using namespace std;

int main()

{

int n, i;

cin>>n;

i = sqrt(n);

for(i ; i>0; i--)

{

if(n%i==0)

{

printf("%d %d\n", min( (n/i), i), max( (n/i), i));

return 0;

}

}

}

<http://codeforces.com/contest/375/problem/A>

A. Divisible by Seven

time limit per test

1 second

memory limit per test

256 megabytes

input

standard input

output

standard output

You have number *a*, whose decimal representation quite luckily contains digits 1, 6, 8, 9. Rearrange the digits in its decimal representation so that the resulting number will be divisible by 7.

Number *a* doesn't contain any leading zeroes and contains digits 1, 6, 8, 9 (it also can contain another digits). The resulting number also mustn't contain any leading zeroes.

**Input**

The first line contains positive integer *a* in the decimal record. It is guaranteed that the record of number *a* contains digits: 1, 6, 8, 9. Number *a* doesn't contain any leading zeroes. The decimal representation of number *a* contains at least 4 and at most 106 characters.

**Output**

Print a number in the decimal notation without leading zeroes — the result of the permutation.

If it is impossible to rearrange the digits of the number *a* in the required manner, print 0.

**Examples**

**input**

1689

**output**

1869

**input**

18906

**output**

18690

<http://codeforces.com/contest/375/problem/D>

D. Tree and Queries

time limit per test

1 second

memory limit per test

256 megabytes

input

standard input

output

standard output

You have a rooted tree consisting of *n* vertices. Each vertex of the tree has some color. We will assume that the tree vertices are numbered by integers from 1 to *n*. Then we represent the color of vertex *v* as *cv*. The tree root is a vertex with number 1.

In this problem you need to answer to *m* queries. Each query is described by two integers *vj*, *kj*. The answer to query *vj*, *kj* is the number of such colors of vertices *x*, that the subtree of vertex *vj* contains at least *kj* vertices of color *x*.

You can find the definition of a rooted tree by the following link: http://en.wikipedia.org/wiki/Tree\_(graph\_theory).

**Input**

The first line contains two integers *n* and *m* (2 ≤ *n* ≤ 105; 1 ≤ *m* ≤ 105). The next line contains a sequence of integers *c*1, *c*2, ..., *cn*(1 ≤ *ci* ≤ 105). The next *n* - 1 lines contain the edges of the tree. The *i*-th line contains the numbers *ai*, *bi* (1 ≤ *ai*, *bi* ≤ *n*; *ai* ≠ *bi*) — the vertices connected by an edge of the tree.

Next *m* lines contain the queries. The *j*-th line contains two integers *vj*, *kj* (1 ≤ *vj* ≤ *n*; 1 ≤ *kj* ≤ 105).

**Output**

Print *m* integers — the answers to the queries in the order the queries appear in the input.

**Examples**

**input**

8 5  
1 2 2 3 3 2 3 3  
1 2  
1 5  
2 3  
2 4  
5 6  
5 7  
5 8  
1 2  
1 3  
1 4  
2 3  
5 3

**output**

2  
2  
1  
0  
1

**input**

4 1  
1 2 3 4  
1 2  
2 3  
3 4  
1 1

**output**

4

**Note**

A subtree of vertex *v* in a rooted tree with root *r* is a set of vertices {*u* : *dist*(*r*, *v*) + *dist*(*v*, *u*) = *dist*(*r*, *u*)}. Where *dist*(*x*, *y*) is the length (in edges) of the shortest path between vertices *x* and *y*.

#include <bits/stdc++.h>

#define fast\_io ios\_base::sync\_with\_stdio(false);cin.tie(0);

#define ll long long

#define MOD 1000000001

#define PI 3.1415926535897

#define eps 0.0000001

#define INF 100000000000000000LL

#define pb push\_back

#define mp make\_pair

#define REP(i,n) for(i=0;i<n;i++)

#define FOR(i,n) for(i=1;i<=n;i++)

#define all(c) c.begin(),c.end()

#define CLEAR(a) memset(a,0,sizeof(a))

#define MINUS(a) memset(a,-1,sizeof(a))

#define pi pair<int,int>

#define pii pair<pi,int>

#define piii pair<pi,pi>

#define F first

#define S second

using namespace std;

int block\_size,timer=0,z[100001];

int cnt[100001],c[100001],col[100001],tin[100001],tout[100001];

bool visited[100001];

vector<int> g[100001];

bool comp(piii a,piii b)

{

if(a.F.F/block\_size != b.F.F/block\_size)

{

return a.F.F/block\_size<b.F.F/block\_size;

}

return a.F.S<b.F.S;

}

void add(int x)

{

cnt[z[x]]++;

c[cnt[z[x]]]++;

}

void remove(int x)

{

c[cnt[z[x]]]--;

cnt[z[x]]--;

}

void dfs(int u)

{

visited[u]=true;

tin[u]=++timer;

for(int i=0;i<g[u].size();i++)

{

int v=g[u][i];

if(!visited[v])

{

dfs(v);

}

}

tout[u]=timer;

}

int main()

{

fast\_io;

int n,i,a,b,l,r,q,lcurr,rcurr,k,ans[100001];

vector<piii> v;

cin>>n>>q;

block\_size=2\*sqrt(n);

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

{

cin>>col[i];

}

for(i=0;i<n-1;i++)

{

cin>>a>>b;

g[a].pb(b);

g[b].pb(a);

}

dfs(1);

for(i=0;i<q;i++)

{

cin>>l>>r;

v.pb(mp(mp(tin[l],tout[l]),mp(i,r)));

}

sort(all(v),comp);

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

{

z[tin[i]]=col[i];

}

lcurr=1;

rcurr=1;

for(i=0;i<q;i++)

{

l=v[i].F.F;

r=v[i].F.S;

k=v[i].S.S;

while(l>lcurr)

{

remove(lcurr);

lcurr++;

}

while(l<lcurr)

{

lcurr--;

add(lcurr);

}

while(r<rcurr-1)

{

remove(rcurr-1);

rcurr--;

}

while(r>=rcurr)

{

add(rcurr);

rcurr++;

}

ans[v[i].S.F]=c[k];

/\*cout<<cnt[1]<<" "<<cnt[2]<<" "<<cnt[3]<<endl;

cout<<c[0]<<" "<<c[1]<<" "<<c[2]<<endl;

cout<<"---";\*/

}

for(i=0;i<q;i++)

{

cout<<ans[i]<<"\n";

}

return 0;

}

<http://codeforces.com/contest/9/problem/A>

A. Die Roll

time limit per test

1 second

memory limit per test

64 megabytes

input

standard input

output

standard output

Yakko, Wakko and Dot, world-famous animaniacs, decided to rest from acting in cartoons, and take a leave to travel a bit. Yakko dreamt to go to Pennsylvania, his Motherland and the Motherland of his ancestors. Wakko thought about Tasmania, its beaches, sun and sea. Dot chose Transylvania as the most mysterious and unpredictable place.

But to their great regret, the leave turned to be very short, so it will be enough to visit one of the three above named places. That's why Yakko, as the cleverest, came up with a truly genius idea: let each of the three roll an ordinary six-sided die, and the one with the highest amount of points will be the winner, and will take the other two to the place of his/her dreams.

Yakko thrown a die and got Y points, Wakko — W points. It was Dot's turn. But she didn't hurry. Dot wanted to know for sure what were her chances to visit Transylvania.

It is known that Yakko and Wakko are true gentlemen, that's why if they have the same amount of points with Dot, they will let Dot win.

**Input**

The only line of the input file contains two natural numbers Y and W — the results of Yakko's and Wakko's die rolls.

**Output**

Output the required probability in the form of irreducible fraction in format «A/B», where A — the numerator, and B — the denominator. If the required probability equals to zero, output «0/1». If the required probability equals to 1, output «1/1».

**Examples**

**input**

4 2

**output**

1/2

**Note**

Dot will go to Transylvania, if she is lucky to roll 4, 5 or 6 points.

#include<bits/stdc++.h>

using namespace std;

int main()

{

int a,b,w,ww;

while(scanf("%d %d",&a,&b)==2)

{

if(a>b)

w=a;

else

w=b;

ww=7-w;

if(ww==2 || ww==4)

printf("%d/3\n", ww/2);

else if(ww==6 || ww==3)

printf("1/%d\n",6/ww);

else

printf("%d/6\n",ww );

}

return 0;

}

<http://codeforces.com/contest/118/problem/B>

B. Present from Lena

time limit per test

2 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

Vasya's birthday is approaching and Lena decided to sew a patterned handkerchief to him as a present. Lena chose digits from 0 to *n*as the pattern. The digits will form a rhombus. The largest digit *n* should be located in the centre. The digits should decrease as they approach the edges. For example, for *n* = 5 the handkerchief pattern should look like that:

          0  
        0 1 0  
      0 1 2 1 0  
    0 1 2 3 2 1 0  
  0 1 2 3 4 3 2 1 0  
0 1 2 3 4 5 4 3 2 1 0  
  0 1 2 3 4 3 2 1 0  
    0 1 2 3 2 1 0  
      0 1 2 1 0  
        0 1 0  
          0

Your task is to determine the way the handkerchief will look like by the given *n*.

**Input**

The first line contains the single integer *n* (2 ≤ *n* ≤ 9).

**Output**

Print a picture for the given *n*. You should strictly observe the number of spaces before the first digit on each line. Every two adjacent digits in the same line should be separated by exactly one space. There should be no spaces after the last digit at the end of each line.

**Examples**

**input**

2

**output**

0  
 0 1 0  
0 1 2 1 0  
 0 1 0  
 0

**input**

3

**output**

0  
 0 1 0  
 0 1 2 1 0  
0 1 2 3 2 1 0  
 0 1 2 1 0  
 0 1 0  
 0

#include <iostream>

using namespace std;

int main()

{

int n;

cin>>n;

for (int i=0;i<=n;i++)

{

if (i>0)

{

for (int j=i;j<n;j++)

cout<<" ";

for (int j=i;j<n;j++)

cout<<" ";

for (int j=0;j<=i;j++)

cout<<j<<" ";

for (int j=0;j<i-1;j++)

cout<<i-j-1<<" ";

cout<<"0";

}

else

{

for (int j=i;j<n;j++)

cout<<" ";

for (int j=i;j<n;j++)

cout<<" ";

cout<<"0";

}

cout<<endl;

}

for (int i=n;i>0;i--)

{

if (i>1)

{

for (int j=i;j<=n;j++)

cout<<" ";

for (int j=i;j<=n;j++)

cout<<" ";

for (int j=0;j<=i-1;j++)

cout<<j<<" ";

for (int j=0;j<i-2;j++)

cout<<i-j-2<<" ";

cout<<"0";

cout<<endl;

}

else

{

for (int j=i;j<=n;j++)

cout<<" ";

for (int j=i;j<=n;j++)

cout<<" ";

cout<<"0";

}

}

return 0;

}

<http://codeforces.com/contest/731/problem/B>

B. Coupons and Discounts

time limit per test

1 second

memory limit per test

256 megabytes

input

standard input

output

standard output

The programming competition season has already started and it's time to train for ICPC. Sereja coaches his teams for a number of year and he knows that to get ready for the training session it's not enough to prepare only problems and editorial. As the training sessions lasts for several hours, teams become hungry. Thus, Sereja orders a number of pizzas so they can eat right after the end of the competition.

Teams plan to train for *n* times during *n* consecutive days. During the training session Sereja orders exactly one pizza for each team that is present this day. He already knows that there will be *ai* teams on the *i*-th day.

There are two types of discounts in Sereja's favourite pizzeria. The first discount works if one buys two pizzas at one day, while the second is a coupon that allows to buy one pizza during two **consecutive** days (two pizzas in total).

As Sereja orders really a lot of pizza at this place, he is the golden client and can use the unlimited number of discounts and coupons of any type at any days.

Sereja wants to order exactly *ai* pizzas on the *i*-th day while using only discounts and coupons. Note, that he will never buy more pizzas than he need for this particular day. Help him determine, whether he can buy the proper amount of pizzas each day if he is allowed to use only coupons and discounts. Note, that it's also prohibited to have any active coupons after the end of the day *n*.

**Input**

The first line of input contains a single integer *n* (1 ≤ *n* ≤ 200 000) — the number of training sessions.

The second line contains *n* integers *a*1, *a*2, ..., *an* (0 ≤ *ai* ≤ 10 000) — the number of teams that will be present on each of the days.

**Output**

If there is a way to order pizzas using only coupons and discounts and do not buy any extra pizzas on any of the days, then print "YES" (without quotes) in the only line of output. Otherwise, print "NO" (without quotes).

**Examples**

**input**

4  
1 2 1 2

**output**

YES

**input**

3  
1 0 1

**output**

NO

**Note**

In the first sample, Sereja can use one coupon to buy one pizza on the first and the second days, one coupon to buy pizza on the second and the third days and one discount to buy pizzas on the fourth days. This is the only way to order pizzas for this sample.

In the second sample, Sereja can't use neither the coupon nor the discount without ordering an extra pizza. Note, that it's possible that there will be no teams attending the training sessions on some days.

#include <iostream>

#include <cmath>

using namespace std;

int main(){

long int n;

cin >> n;

int a[n],bedehi = 0;

for(int i = 0;i < n;i++){

cin >> a[i];

}

for(int i = 0;i < n ;i++){

a[i] += bedehi;

a[i] %= 2;

if(a[i] != 0){

bedehi = -1;

a[i]--;

}

else{

bedehi = 0;

}

}

for(int i = 0;i < n ;i++){

if(a[i] != 0 || bedehi != 0){

cout << "NO";

return 0;

}

}

cout << "Yes";

return 0;

}

<http://codeforces.com/contest/339/problem/A>

A. Helpful Maths

time limit per test

2 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

Xenia the beginner mathematician is a third year student at elementary school. She is now learning the addition operation.

The teacher has written down the sum of multiple numbers. Pupils should calculate the sum. To make the calculation easier, the sum only contains numbers 1, 2 and 3. Still, that isn't enough for Xenia. She is only beginning to count, so she can calculate a sum only if the summands follow in non-decreasing order. For example, she can't calculate sum 1+3+2+1 but she can calculate sums 1+1+2 and 3+3.

You've got the sum that was written on the board. Rearrange the summans and print the sum in such a way that Xenia can calculate the sum.

**Input**

The first line contains a non-empty string *s* — the sum Xenia needs to count. String *s* contains no spaces. It only contains digits and characters "+". Besides, string *s* is a correct sum of numbers 1, 2 and 3. String *s* is at most 100 characters long.

**Output**

Print the new sum that Xenia can count.

**Examples**

**input**

3+2+1

**output**

1+2+3

**input**

1+1+3+1+3

**output**

1+1+1+3+3

**input**

2

**output**

2

#include<bits/stdc++.h>

using namespace std;

int main()

{

char a[110],ans[110],temp;

int i,a1=0,a2=0,a3=0;

temp=a[0];

cin>>a;

for(i=0;i<=strlen(a);i+=2){

if(a[i]=='1') a1++;

else if(a[i]=='2') a2++;

else if(a[i]=='3') a3++;

}

for(i=0;i<2\*a1;i+=2){a[i]='1';}

for(i=2\*a1;i<2\*(a1+a2);i+=2){a[i]='2';}

for(i=2\*(a1+a2);i<2\*(a1+a2+a3);i+=2){a[i]='3';}

cout<<a;

return 0;

}

<http://codeforces.com/contest/432/problem/A>

A. Choosing Teams

time limit per test

1 second

memory limit per test

256 megabytes

input

standard input

output

standard output

The Saratov State University Olympiad Programmers Training Center (SSU OPTC) has *n* students. For each student you know the number of times he/she has participated in the ACM ICPC world programming championship. According to the ACM ICPC rules, each person can participate in the world championship at most 5 times.

The head of the SSU OPTC is recently gathering teams to participate in the world championship. Each team must consist of exactly three people, at that, any person cannot be a member of two or more teams. What maximum number of teams can the head make if he wants each team to participate in the world championship with the same members at least *k* times?

**Input**

The first line contains two integers, *n* and *k* (1 ≤ *n* ≤ 2000; 1 ≤ *k* ≤ 5). The next line contains *n* integers: *y*1, *y*2, ..., *yn* (0 ≤ *yi* ≤ 5), where *yi* shows the number of times the *i*-th person participated in the ACM ICPC world championship.

**Output**

Print a single number — the answer to the problem.

**Examples**

**input**

5 2  
0 4 5 1 0

**output**

1

**input**

6 4  
0 1 2 3 4 5

**output**

0

**input**

6 5  
0 0 0 0 0 0

**output**

2

**Note**

In the first sample only one team could be made: the first, the fourth and the fifth participants.

In the second sample no teams could be created.

In the third sample two teams could be created. Any partition into two teams fits.

#include<stdio.h>

#include<algorithm>

using namespace std;

int main()

{

int n,k,sum=0,cnt=0,i,j,p=0;

scanf("%d%d",&n,&k);

int a[n];

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

{

scanf("%d",&a[i]);

}

sort(a+1,a+n+1);

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

{

sum=a[j]+k;

//printf("djfh");

if(sum<=5)

{

p++;

sum=0;

}

else

break;

if(p==3)

{

cnt++;

p=0;

}

}

printf("%d\n",cnt);

return 0;

}

<http://codeforces.com/contest/672/problem/A>

A. Summer Camp

time limit per test

1 second

memory limit per test

256 megabytes

input

standard input

output

standard output

Every year, hundreds of people come to summer camps, they learn new algorithms and solve hard problems.

This is your first year at summer camp, and you are asked to solve the following problem. All integers starting with 1 are written in one line. The prefix of these line is "123456789101112131415...". Your task is to print the *n*-th digit of this string (digits are numbered starting with 1.

**Input**

The only line of the input contains a single integer *n* (1 ≤ *n* ≤ 1000) — the position of the digit you need to print.

**Output**

Print the *n*-th digit of the line.

**Examples**

**input**

3

**output**

3

**input**

11

**output**

0

**Note**

In the first sample the digit at position 3 is '3', as both integers 1 and 2 consist on one digit.

In the second sample, the digit at position 11 is '0', it belongs to the integer 10.

<http://codeforces.com/contest/651/problem/A>

A. Joysticks

time limit per test

1 second

memory limit per test

256 megabytes

input

standard input

output

standard output

Friends are going to play console. They have two joysticks and only one charger for them. Initially first joystick is charged at *a*1 percent and second one is charged at *a*2 percent. You can connect charger to a joystick only at the beginning of each minute. In one minute joystick either discharges by 2 percent (if not connected to a charger) or charges by 1 percent (if connected to a charger).

Game continues while both joysticks have a positive charge. Hence, if at the beginning of minute some joystick is charged by 1 percent, it has to be connected to a charger, otherwise the game stops. If some joystick completely discharges (its charge turns to 0), the game also stops.

Determine the maximum number of minutes that game can last. It is prohibited to pause the game, i. e. at each moment both joysticks should be enabled. It is allowed for joystick to be charged by **more than 100 percent**.

**Input**

The first line of the input contains two positive integers *a*1 and *a*2 (1 ≤ *a*1, *a*2 ≤ 100), the initial charge level of first and second joystick respectively.

**Output**

Output the only integer, the maximum number of minutes that the game can last. Game continues until some joystick is discharged.

**Examples**

**input**

3 5

**output**

6

**input**

4 4

**output**

5

**Note**

In the first sample game lasts for 6 minute by using the following algorithm:

* at the beginning of the first minute connect first joystick to the charger, by the end of this minute first joystick is at 4%, second is at 3%;
* continue the game without changing charger, by the end of the second minute the first joystick is at 5%, second is at 1%;
* at the beginning of the third minute connect second joystick to the charger, after this minute the first joystick is at 3%, the second one is at 2%;
* continue the game without changing charger, by the end of the fourth minute first joystick is at 1%, second one is at 3%;
* at the beginning of the fifth minute connect first joystick to the charger, after this minute the first joystick is at 2%, the second one is at 1%;
* at the beginning of the sixth minute connect second joystick to the charger, after this minute the first joystick is at 0%, the second one is at 2%.

After that the first joystick is completely discharged and the game is stopped.

#include<stdio.h>

#include<algorithm>

using namespace std;

int main()

{

int n,k,sum=0,cnt=0,i,j,p=0;

scanf("%d%d",&n,&k);

int a[n];

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

{

scanf("%d",&a[i]);

}

sort(a+1,a+n+1);

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

{

sum=a[j]+k;

//printf("djfh");

if(sum<=5)

{

p++;

sum=0;

}

else

break;

if(p==3)

{

cnt++;

p=0;

}

}

printf("%d\n",cnt);

return 0;

}

<http://codeforces.com/contest/702/problem/B>

B. Powers of Two

time limit per test

3 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

You are given *n* integers *a*1, *a*2, ..., *an*. Find the number of pairs of indexes *i*, *j* (*i* < *j*) that *ai* + *aj* is a power of 2 (i. e. some integer *x*exists so that *ai* + *aj* = 2*x*).

**Input**

The first line contains the single positive integer *n* (1 ≤ *n* ≤ 105) — the number of integers.

The second line contains *n* positive integers *a*1, *a*2, ..., *an* (1 ≤ *ai* ≤ 109).

**Output**

Print the number of pairs of indexes *i*, *j* (*i* < *j*) that *ai* + *aj* is a power of 2.

**Examples**

**input**

4  
7 3 2 1

**output**

2

**input**

3  
1 1 1

**output**

3

**Note**

In the first example the following pairs of indexes include in answer: (1, 4) and (2, 4).

In the second example all pairs of indexes (*i*, *j*) (where *i* < *j*) include in answer.

#include <iostream>

#include <cstdio>

#include <map>

using namespace std;

map <int,int> g;

long long a[200500],ans;

int main()

{

int n;

cin>>n;

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

cin>>a[i];

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

{

for (int j=0; j<31; j++)

{

long long k=(1LL<<j);

ans+=g[k-a[i]];

}

g[a[i]]++;

}

cout<<ans;

return 0;

}

<http://codeforces.com/contest/71/problem/A>

A. Way Too Long Words

time limit per test

2 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

Sometimes some words like "localization" or "internationalization" are so long that writing them many times in one text is quite tiresome.

Let's consider a word *too long*, if its length is **strictly more** than 10 characters. All too long words should be replaced with a special abbreviation.

This abbreviation is made like this: we write down the first and the last letter of a word and between them we write the number of letters between the first and the last letters. That number is in decimal system and doesn't contain any leading zeroes.

Thus, "localization" will be spelt as "l10n", and "internationalization» will be spelt as "i18n".

You are suggested to automatize the process of changing the words with abbreviations. At that all too long words should be replaced by the abbreviation and the words that are not too long should not undergo any changes.

**Input**

The first line contains an integer *n* (1 ≤ *n* ≤ 100). Each of the following *n* lines contains one word. All the words consist of lowercase Latin letters and possess the lengths of from 1 to 100 characters.

**Output**

Print *n* lines. The *i*-th line should contain the result of replacing of the *i*-th word from the input data.

**Examples**

**input**

4  
word  
localization  
internationalization  
pneumonoultramicroscopicsilicovolcanoconiosis

**output**

word  
l10n  
i18n  
p43s

#include <iostream>

#include<cstring>

using namespace std;

int main(){

int j,n;

char a[100];

cin>>n;

while(n--){

cin>>a;

j=strlen(a);

if(j>10){

cout<<a[0]<<j-2<<a[j-1]<<endl;

}

else{

cout<<a<<endl;

}

}

return 0;

}

<http://codeforces.com/contest/266/problem/B>

B. Queue at the School

time limit per test

2 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

During the break the schoolchildren, boys and girls, formed a queue of *n* people in the canteen. Initially the children stood in the order they entered the canteen. However, after a while the boys started feeling awkward for standing in front of the girls in the queue and they started letting the girls move forward each second.

Let's describe the process more precisely. Let's say that the positions in the queue are sequentially numbered by integers from 1 to *n*, at that the person in the position number 1 is served first. Then, if at time *x* a boy stands on the *i*-th position and a girl stands on the (*i* + 1)-th position, then at time *x* + 1 the *i*-th position will have a girl and the (*i* + 1)-th position will have a boy. The time is given in seconds.

You've got the initial position of the children, at the initial moment of time. Determine the way the queue is going to look after *t* seconds.

**Input**

The first line contains two integers *n* and *t* (1 ≤ *n*, *t* ≤ 50), which represent the number of children in the queue and the time after which the queue will transform into the arrangement you need to find.

The next line contains string *s*, which represents the schoolchildren's initial arrangement. If the *i*-th position in the queue contains a boy, then the *i*-th character of string *s* equals "B", otherwise the *i*-th character equals "G".

**Output**

Print string *a*, which describes the arrangement after *t* seconds. If the *i*-th position has a boy after the needed time, then the *i*-th character *a* must equal "B", otherwise it must equal "G".

**Examples**

**input**

5 1  
BGGBG

**output**

GBGGB

**input**

5 2  
BGGBG

**output**

GGBGB

**input**

4 1  
GGGB

**output**

GGGB

#include <iostream>

#include <string>

using namespace std;

int main()

{

int a, b, c, d, e;

cin>>a>>b;

char line[50];

for(c=0; c<a; c++)

cin>>line[c];

for(d=0; d<b; d++)

{

for(e=0; e<a-1; e++)

{

if(line[e]=='B' && line[e+1]=='G')

{

swap(line[e], line[e+1]);

e++;

}

}

}

for(d=0; d<a; d++)

cout<<line[d];

return 0;

}

<http://codeforces.com/contest/227/problem/D>

D. Naughty Stone Piles

time limit per test

2 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

There are *n* piles of stones of sizes *a*1, *a*2, ..., *an* lying on the table in front of you.

During one move you can take one pile and add it to the other. As you add pile *i* to pile *j*, the size of pile *j* increases by the current size of pile *i*, and pile *i* stops existing. The cost of the adding operation equals the size of the added pile.

Your task is to determine the minimum cost at which you can gather all stones in one pile.

To add some challenge, the stone piles built up conspiracy and decided that each pile will let you add to it not more than *k* times (after that it can only be added to another pile).

Moreover, the piles decided to puzzle you completely and told you *q* variants (not necessarily distinct) of what *k* might equal.

Your task is to find the minimum cost for each of *q* variants.

**Input**

The first line contains integer *n* (1 ≤ *n* ≤ 105) — the number of stone piles. The second line contains *n* space-separated integers: *a*1, *a*2, ..., *an* (1 ≤ *ai* ≤ 109) — the initial sizes of the stone piles.

The third line contains integer *q* (1 ≤ *q* ≤ 105) — the number of queries. The last line contains *q* space-separated integers *k*1, *k*2, ..., *kq*(1 ≤ *ki* ≤ 105) — the values of number *k* for distinct queries. Note that numbers *ki* can repeat.

**Output**

Print *q* whitespace-separated integers — the answers to the queries in the order, in which the queries are given in the input.

Please, do not use the %lld specifier to read or write 64-bit integers in C++. It is preferred to use the cin, cout streams or the %I64dspecifier.

**Examples**

**input**

5  
2 3 4 1 1  
2  
2 3

**output**

9 8

**Note**

In the first sample one way to get the optimal answer goes like this: we add in turns the 4-th and the 5-th piles to the 2-nd one; then we add the 1-st pile to the 3-rd one; we add the 2-nd pile to the 3-rd one. The first two operations cost 1 each; the third one costs 2, the fourth one costs 5 (the size of the 2-nd pile after the first two operations is not 3, it already is 5).

In the second sample you can add the 2-nd pile to the 3-rd one (the operations costs 3); then the 1-st one to the 3-th one (the cost is 2); then the 5-th one to the 4-th one (the costs is 1); and at last, the 4-th one to the 3-rd one (the cost is 2).

<http://codeforces.com/contest/727/problem/D>

D. T-shirts Distribution

time limit per test

1 second

memory limit per test

256 megabytes

input

standard input

output

standard output

The organizers of a programming contest have decided to present t-shirts to participants. There are six different t-shirts sizes in this problem: S, M, L, XL, XXL, XXXL (sizes are listed in increasing order). The t-shirts are already prepared. For each size from S to XXXLyou are given the number of t-shirts of this size.

During the registration, the organizers asked each of the *n* participants about the t-shirt size he wants. If a participant hesitated between two sizes, he could specify two neighboring sizes — this means that any of these two sizes suits him.

Write a program that will determine whether it is possible to present a t-shirt to each participant of the competition, or not. Of course, each participant should get a t-shirt of proper size:

* the size he wanted, if he specified one size;
* any of the two neibouring sizes, if he specified two sizes.

If it is possible, the program should find any valid distribution of the t-shirts.

**Input**

The first line of the input contains six non-negative integers — the number of t-shirts of each size. The numbers are given for the sizes S, M, L, XL, XXL, XXXL, respectively. The total number of t-shirts doesn't exceed 100 000.

The second line contains positive integer *n* (1 ≤ *n* ≤ 100 000) — the number of participants.

The following *n* lines contain the sizes specified by the participants, one line per participant. The *i*-th line contains information provided by the *i*-th participant: single size or two sizes separated by comma (without any spaces). If there are two sizes, the sizes are written in increasing order. It is guaranteed that two sizes separated by comma are neighboring.

**Output**

If it is not possible to present a t-shirt to each participant, print «NO» (without quotes).

Otherwise, print *n* + 1 lines. In the first line print «YES» (without quotes). In the following *n* lines print the t-shirt sizes the orginizers should give to participants, one per line. The order of the participants should be the same as in the input.

If there are multiple solutions, print any of them.

**Examples**

**input**

0 1 0 1 1 0  
3  
XL  
S,M  
XL,XXL

**output**

YES  
XL  
M  
XXL

**input**

1 1 2 0 1 1  
5  
S  
M  
S,M  
XXL,XXXL  
XL,XXL

**output**

NO

#include<bits/stdc++.h>

using namespace std;

#define fs first

#define sc second

#define MAX 100000

#define pb push\_back

#define mp make\_pair

#define INF (1LL<<61)

#define MOD 1000000007

typedef long long Int;

typedef pair<Int,Int> pii;

typedef vector<Int> vi;

typedef vector<pii> vii;

Int cnt[6]={0};

string ans[MAX+5];

map<string,Int>M={{"S",0},{"M",1},{"L",2},{"XL",3},{"XXL",4},{"XXXL",5}};

string give(Int x)

{

for (auto it=M.begin();it!=M.end();++it)

{

if (it->sc==x)

return it->fs;

}

}

struct par

{

Int c1,c2,id;

par()

{

c1=-1;

c2=-1;

}

}P[MAX+5];

bool cmp(par a,par b)

{

if (a.c1!=b.c1)

return a.c1<b.c1;

else

return a.c2<b.c2;

}

int main()

{

for (Int i=0;i<6;++i)

cin>>cnt[i];

Int n;

cin>>n;

for (Int i=0;i<n;++i)

{

string S,A,B;

cin>>S;

for (Int j=0;j<S.size();++j)

{

if (S[j]==',')

{

A=S.substr(0,j);

B=S.substr(j+1,S.size()-j-1);

}

}

if (A.size()>0)

{

P[i].c1=M[A];

P[i].c2=M[B];

}

else

P[i].c1=M[S];

P[i].id=i;

}

sort(P,P+n,cmp);

bool flag=1;

for (Int i=0;i<n;++i)

{

if (cnt[P[i].c1])

{

--cnt[P[i].c1];

ans[P[i].id]=give(P[i].c1);

}

else if (cnt[P[i].c2])

{

--cnt[P[i].c2];

ans[P[i].id]=give(P[i].c2);

}

else

{

flag=0;

break;

}

}

if (flag)

{

cout<<"YES\n";

for (Int i=0;i<n;++i)

cout<<ans[i]<<"\n";

}

else

cout<<"NO\n";

return 0;

}

<http://codeforces.com/contest/706/problem/C>

C. Hard problem

time limit per test

1 second

memory limit per test

256 megabytes

input

standard input

output

standard output

Vasiliy is fond of solving different tasks. Today he found one he wasn't able to solve himself, so he asks you to help.

Vasiliy is given *n* strings consisting of lowercase English letters. He wants them to be sorted in lexicographical order (as in the dictionary), but he is not allowed to swap any of them. The only operation he is allowed to do is to reverse any of them (first character becomes last, second becomes one before last and so on).

To reverse the *i*-th string Vasiliy has to spent *ci* units of energy. He is interested in the minimum amount of energy he has to spent in order to have strings sorted in lexicographical order.

String *A* is lexicographically smaller than string *B* if it is shorter than *B* (|*A*| < |*B*|) and is its prefix, or if none of them is a prefix of the other and at the first position where they differ character in *A* is smaller than the character in *B*.

For the purpose of this problem, two equal strings nearby do not break the condition of sequence being sorted lexicographically.

**Input**

The first line of the input contains a single integer *n* (2 ≤ *n* ≤ 100 000) — the number of strings.

The second line contains *n* integers *ci* (0 ≤ *ci* ≤ 109), the *i*-th of them is equal to the amount of energy Vasiliy has to spent in order to reverse the *i*-th string.

Then follow *n* lines, each containing a string consisting of lowercase English letters. The total length of these strings doesn't exceed 100 000.

**Output**

If it is impossible to reverse some of the strings such that they will be located in lexicographical order, print  - 1. Otherwise, print the minimum total amount of energy Vasiliy has to spent.

**Examples**

**input**

2  
1 2  
ba  
ac

**output**

1

**input**

3  
1 3 1  
aa  
ba  
ac

**output**

1

**input**

2  
5 5  
bbb  
aaa

**output**

-1

**input**

2  
3 3  
aaa  
aa

**output**

-1

**Note**

In the second sample one has to reverse string 2 or string 3. To amount of energy required to reverse the string 3 is smaller.

In the third sample, both strings do not change after reverse and they go in the wrong order, so the answer is  - 1.

In the fourth sample, both strings consists of characters 'a' only, but in the sorted order string "aa" should go before string "aaa", thus the answer is  - 1.

#include <bits/stdc++.h>

#define ll long long

#define fr first

#define sc second

#define mp make\_pair

#define pb push\_back

#define LEFT(a) ((a)<<1)

#define RIGHT(a) (LEFT(a) + 1)

#define MID(a,b) ((a+b)>>1)

using namespace std;

const int MOD=1e9+7,N=100005;

const ll INF=1e11;

int n, c[N];

ll dpy[N], dpn[N];

string s[N],ss[N];

string f (string str)

{

string sss;

for (int i=str.size()-1;i>=0;i--)sss+=str[i];

return sss;

}

int main()

{

cin>>n;

for (int i=1;i<=n;i++)cin>>c[i];

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

cin>>s[i];

ss[i]=f(s[i]);

}

dpy[1]=c[1];

dpn[1]=0;

for (int i=2;i<=n;i++){

dpy[i]=-1;

dpn[i]=-1;

if (s[i]>=s[i-1] && dpn[i-1]!=-1)

dpn[i]=dpn[i-1];

if (s[i]>=ss[i-1] && dpy[i-1]!=-1)

if (dpn[i]==-1 || dpn[i]>dpy[i-1])

dpn[i]=dpy[i-1];

if (ss[i]>=s[i-1] && dpn[i-1]!=-1)

dpy[i]=dpn[i-1]+c[i];

if (ss[i]>=ss[i-1] && dpy[i-1]!=-1)

if (dpy[i]==-1 || dpy[i]>dpy[i-1]+c[i])

dpy[i]=dpy[i-1]+c[i];

}

if (dpy[n]!=-1 && dpn[n]!=-1)cout<<min(dpy[n],dpn[n])<<endl;

else cout<<dpy[n]+dpn[n]+1<<endl;

return 0;

}

<http://codeforces.com/contest/743/problem/D>

D. Chloe and pleasant prizes

time limit per test

2 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

Generous sponsors of the olympiad in which Chloe and Vladik took part allowed all the participants to choose a prize for them on their own. Christmas is coming, so sponsors decided to decorate the Christmas tree with their prizes.

They took *n* prizes for the contestants and wrote on each of them a unique id (integer from 1 to *n*). A gift *i* is characterized by integer *ai* — pleasantness of the gift. The pleasantness of the gift can be positive, negative or zero. Sponsors placed the gift 1 on the top of the tree. All the other gifts hung on a rope tied to some other gift so that each gift hung on the first gift, possibly with a sequence of ropes and another gifts. Formally, the gifts formed a rooted tree with *n* vertices.

The prize-giving procedure goes in the following way: the participants come to the tree one after another, choose any of the remaining gifts and cut the rope this prize hang on. Note that all the ropes which were used to hang other prizes on the chosen one are not cut. So the contestant gets the chosen gift as well as the all the gifts that hang on it, possibly with a sequence of ropes and another gifts.

Our friends, Chloe and Vladik, shared the first place on the olympiad and they will choose prizes at the same time! To keep themselves from fighting, they decided to choose two different gifts so that the sets of the gifts that hang on them with a sequence of ropes and another gifts don't intersect. In other words, there shouldn't be any gift that hang both on the gift chosen by Chloe and on the gift chosen by Vladik. From all of the possible variants they will choose such pair of prizes that the sum of pleasantness of all the gifts that they will take after cutting the ropes is as large as possible.

Print the maximum sum of pleasantness that Vladik and Chloe can get. If it is impossible for them to choose the gifts without fighting, print Impossible.

**Input**

The first line contains a single integer *n* (1 ≤ *n* ≤ 2·105) — the number of gifts.

The next line contains *n* integers *a*1, *a*2, ..., *an* ( - 109 ≤ *ai* ≤ 109) — the pleasantness of the gifts.

The next (*n* - 1) lines contain two numbers each. The *i*-th of these lines contains integers *ui* and *vi* (1 ≤ *ui*, *vi* ≤ *n*, *ui* ≠ *vi*) — the description of the tree's edges. It means that gifts with numbers *ui* and *vi* are connected to each other with a rope. The gifts' ids in the description of the ropes can be given in arbirtary order: *vi* hangs on *ui* or *ui* hangs on *vi*.

It is guaranteed that all the gifts hang on the first gift, possibly with a sequence of ropes and another gifts.

**Output**

If it is possible for Chloe and Vladik to choose prizes without fighting, print single integer — the maximum possible sum of pleasantness they can get together.

Otherwise print Impossible.

**Examples**

**input**

8  
0 5 -1 4 3 2 6 5  
1 2  
2 4  
2 5  
1 3  
3 6  
6 7  
6 8

**output**

25

**input**

4  
1 -5 1 1  
1 2  
1 4  
2 3

**output**

2

**input**

1  
-1

**output**

Impossible

#include <bits/stdc++.h>

using namespace std;

#ifndef ONLINE\_JUDGE

#include "../debug.hpp"

struct debugger dbg;

#else

#define debug(args...) // Just strip off all debug tokens

#endif

#define si(i) scanf("%d",&i)

#define si2(i,j) scanf("%d %d",&i,&j)

#define si3(i,j,k) scanf("%d %d %d",&i,&j,&k)

#define slli(i) scanf("%I64d",&i)

#define slli2(i,j) scanf("%I64d %I64d",&i,&j)

#define slli3(i,j,k) scanf("%I64d %I64d %I64d",&i,&j,&k)

#define pi(i) printf("%d\n",i)

#define plli(i) printf("%I64d\n",i)

#define forup(i,a,b) for(int i = (a); (i) < (b); ++(i))

#define fordn(i,a,b) for(int i = (a); (i) > (b); --(i))

#define rep(i,a) for(int i = 0; (i) < (a); ++(i))

#define SYNC ios\_base::sync\_with\_stdio(0)

#define mp make\_pair

#define FF first

#define SS second

#define pb push\_back

#define fill(a,v) memset(a,v,sizeof a)

#define ceil(a,b) (((a)%(b)==0)?((a)/(b)):((a)/(b)+1))

#define rem(a,b) ((a<0)?((((a)%(b))+(b))%(b)):((a)%(b)))

#define MOD 1000000007LL

#define INF INT\_MAX

#define N 200007

typedef long long int ll;

typedef pair<int,int> PII;

typedef pair<ll,ll> PLL;

typedef vector<string> VS;

typedef vector<int> VI;

typedef vector<ll> VL;

typedef vector<PII> VOII;

typedef vector<PLL> VOLL;

typedef vector<VI> VOVI;

int dX[] = {-1,0,1,0,-1,1,1,-1};

int dY[] = {0,1,0,-1,1,1,-1,-1};

ll n;

ll a[N];

VL adjList[N];

bool isPossible;

ll ans;

void preProcess(ll cur,ll par) {

for (ll i = 0; i < adjList[cur].size(); ++i) {

ll nxt = adjList[cur][i];

if(nxt != par) {

preProcess(nxt,cur);

a[cur] += a[nxt];

}

}

return;

}

ll dfs(ll cur,ll par) {

VL tmp;

ll max\_in\_this = a[cur];

for (ll i = 0; i < adjList[cur].size(); ++i) {

ll nxt = adjList[cur][i];

if(nxt != par) tmp.pb(dfs(nxt,cur));

}

if(tmp.size() > 0) {

sort(tmp.begin(), tmp.end());

max\_in\_this = max(tmp.back(),max\_in\_this);

}

if(tmp.size() > 1) {

isPossible = true;

ll max\_pair\_sum = tmp.back();

tmp.pop\_back();

max\_pair\_sum += tmp.back();

ans = max(ans,max\_pair\_sum);

}

return max\_in\_this;

}

int main()

{

ll u,v;

cin >> n;

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

cin >> a[i];

}

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

cin >> u >> v;

adjList[u].pb(v);

adjList[v].pb(u);

}

isPossible = false;

ans = MOD\*(10000000LL)\*(-1);

preProcess(1,-1);

dfs(1,-1);

if(!isPossible) cout << "Impossible" << endl;

else cout << ans << endl;

return 0;

}

<http://codeforces.com/contest/1/problem/A>

A. Theatre Square

time limit per test

1 second

memory limit per test

256 megabytes

input

standard input

output

standard output

Theatre Square in the capital city of Berland has a rectangular shape with the size *n* × *m* meters. On the occasion of the city's anniversary, a decision was taken to pave the Square with square granite flagstones. Each flagstone is of the size *a* × *a*.

What is the least number of flagstones needed to pave the Square? It's allowed to cover the surface larger than the Theatre Square, but the Square has to be covered. It's not allowed to break the flagstones. The sides of flagstones should be parallel to the sides of the Square.

**Input**

The input contains three positive integer numbers in the first line: *n*,  *m* and *a* (1 ≤  *n*, *m*, *a* ≤ 109).

**Output**

Write the needed number of flagstones.

**Examples**

**input**

6 6 4

**output**

4

#include<iostream>

using namespace std;

int main()

{long long int m,n,a,s,d,f;

cin>>m>>n>>a;

if(m%a==0)

{ s=m/a;}

else

{s=(m/a)+1;}

if(n%a==0)

{d=n/a;}

else

{d=(n/a)+1;}

f=s\*d;

cout<<f;

return 0;

}

<http://codeforces.com/contest/630/problem/D>

D. Hexagons!

time limit per test

0.5 seconds

memory limit per test

64 megabytes

input

standard input

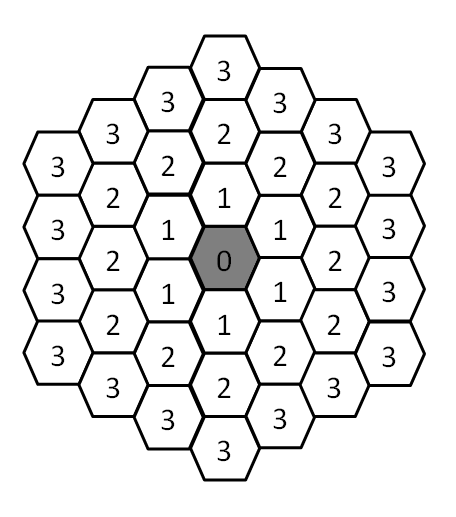
output

standard output

After a probationary period in the game development company of IT City Petya was included in a group of the programmers that develops a new turn-based strategy game resembling the well known "Heroes of Might & Magic". A part of the game is turn-based fights of big squadrons of enemies on infinite fields where every cell is in form of a hexagon.

Some of magic effects are able to affect several field cells at once, cells that are situated not farther than *n* cells away from the cell in which the effect was applied. The distance between cells is the minimum number of cell border crosses on a path from one cell to another.

It is easy to see that the number of cells affected by a magic effect grows rapidly when *n* increases, so it can adversely affect the game performance. That's why Petya decided to write a program that can, given *n*, determine the number of cells that should be repainted after effect application, so that game designers can balance scale of the effects and the game performance. Help him to do it. Find the number of hexagons situated not farther than *n* cells away from a given cell.



**Input**

The only line of the input contains one integer *n* (0 ≤ *n* ≤ 109).

**Output**

Output one integer — the number of hexagons situated not farther than *n* cells away from a given cell.

**Examples**

**input**

2

**output**

19

#include<bits/stdc++.h>

using namespace std;

typedef long long unsigned llu;

typedef long long ll;

char a[1000005];

int main()

{

llu n;

cin>>n;

cout<<(((n+1)\*3)\*n)+1;

}

<http://codeforces.com/contest/749/problem/C>

C. Voting

time limit per test

1 second

memory limit per test

256 megabytes

input

standard input

output

standard output

There are *n* employees in Alternative Cake Manufacturing (ACM). They are now voting on some very important question and the leading world media are trying to predict the outcome of the vote.

Each of the employees belongs to one of two fractions: depublicans or remocrats, and these two fractions have opposite opinions on what should be the outcome of the vote. The voting procedure is rather complicated:

1. Each of *n* employees makes a statement. They make statements one by one starting from employees 1 and finishing with employee *n*. If at the moment when it's time for the *i*-th employee to make a statement he no longer has the right to vote, he just skips his turn (and no longer takes part in this voting).
2. When employee makes a statement, he can do nothing or declare that one of the other employees no longer has a right to vote. It's allowed to deny from voting people who already made the statement or people who are only waiting to do so. If someone is denied from voting he no longer participates in the voting till the very end.
3. When all employees are done with their statements, the procedure repeats: again, each employees starting from 1 and finishing with *n* who are still eligible to vote make their statements.
4. The process repeats until there is only one employee eligible to vote remaining and he determines the outcome of the whole voting. Of course, he votes for the decision suitable for his fraction.

You know the order employees are going to vote and that they behave optimal (and they also know the order and who belongs to which fraction). Predict the outcome of the vote.

**Input**

The first line of the input contains a single integer *n* (1 ≤ *n* ≤ 200 000) — the number of employees.

The next line contains *n* characters. The *i*-th character is 'D' if the *i*-th employee is from depublicans fraction or 'R' if he is from remocrats.

**Output**

Print 'D' if the outcome of the vote will be suitable for depublicans and 'R' if remocrats will win.

**Examples**

**input**

5  
DDRRR

**output**

D

**input**

6  
DDRRRR

**output**

R

**Note**

Consider one of the voting scenarios for the first sample:

1. Employee 1 denies employee 5 to vote.
2. Employee 2 denies employee 3 to vote.
3. Employee 3 has no right to vote and skips his turn (he was denied by employee 2).
4. Employee 4 denies employee 2 to vote.
5. Employee 5 has no right to vote and skips his turn (he was denied by employee 1).
6. Employee 1 denies employee 4.
7. Only employee 1 now has the right to vote so the voting ends with the victory of depublicans.

#include<bits/stdc++.h>

using namespace std;

#define ll long long

#define mp make\_pair

#define pb push\_back

int main()

{

ll n;

cin>>n;

string s;

cin>>s;

queue < ll > qd,qr;

for(ll i=0;i<n;i++)

{

if(s[i]=='D')

qd.push(i);

else

qr.push(i);

}

while((qr.size()!=0) && (qd.size()!=0))

{

if(qr.front()<qd.front())

{

qd.pop();

qr.push(qr.front()+n);

qr.pop();

}

else

{

qr.pop();

qd.push(qd.front()+n);

qd.pop();

}

}

if(qr.size()!=0)

cout<<'R';

else

cout<<'D';

return 0;

}

<http://codeforces.com/contest/752/problem/A>

A. Santa Claus and a Place in a Class

time limit per test

2 seconds

memory limit per test

256 megabytes

input

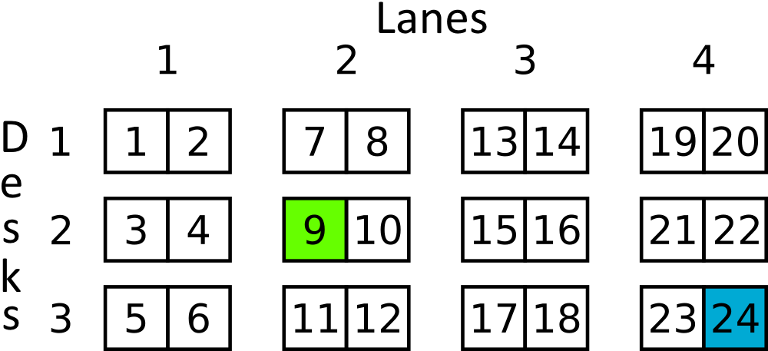
standard input

output

standard output

Santa Claus is the first who came to the Christmas Olympiad, and he is going to be the first to take his place at a desk! In the classroom there are *n* lanes of *m* desks each, and there are two working places at each of the desks. The lanes are numbered from 1 to *n* from the left to the right, the desks in a lane are numbered from 1 to *m* starting from the blackboard. Note that the lanes go perpendicularly to the blackboard, not along it (see picture).

The organizers numbered all the working places from 1 to 2*nm*. The places are numbered by lanes (i. e. all the places of the first lane go first, then all the places of the second lane, and so on), in a lane the places are numbered starting from the nearest to the blackboard (i. e. from the first desk in the lane), at each desk, the place on the left is numbered before the place on the right.

The picture illustrates the first and the second samples.

Santa Clause knows that his place has number *k*. Help him to determine at which lane at which desk he should sit, and whether his place is on the left or on the right!

**Input**

The only line contains three integers *n*, *m* and *k* (1 ≤ *n*, *m* ≤ 10 000, 1 ≤ *k* ≤ 2*nm*) — the number of lanes, the number of desks in each lane and the number of Santa Claus' place.

**Output**

Print two integers: the number of lane *r*, the number of desk *d*, and a character *s*, which stands for the side of the desk Santa Claus. The character *s* should be "L", if Santa Clause should sit on the left, and "R" if his place is on the right.

**Examples**

**input**

4 3 9

**output**

2 2 L

**input**

4 3 24

**output**

4 3 R

**input**

2 4 4

**output**

1 2 R

**Note**

The first and the second samples are shown on the picture. The green place corresponds to Santa Claus' place in the first example, the blue place corresponds to Santa Claus' place in the second example.

In the third sample there are two lanes with four desks in each, and Santa Claus has the fourth place. Thus, his place is in the first lane at the second desk on the right.

#include <bits/stdc++.h>

using namespace std;

typedef long long ll;

typedef vector<int> vi;

typedef vector<vi> vvi;

typedef pair<int,int> ii;

typedef vector<ii> vii;

#define sz(a) int((a).size())

#define mp make\_pair

#define pb push\_back

#define fi first

#define se second

#define buli(x) (\_\_builtin\_popcountll(x))

#define all(c) (c).begin(),(c).end()

#define foreach(i, c) for(\_\_typeof((c).begin()) i = (c).begin(); i != (c).end(); ++i)

#define present(c,x) ((c).find(x) != (c).end())

#define cpresent(c,x) (find(all(c),x) != (c).end())

#define input(s, n) for(int i=0;i<n;i++) cin >> s[i];

#define F(i, f, t) for(int i=f;i<t;i++)

#define R(i,n) for(ll i=0;i<n;i++)

#define print(cnt,x) cout<<"Case #"<<cnt<<": "<<x<<"\n";

template <class T> inline void smax(T &x,T y){ x = max((x), (y));}

template <class T> inline void smin(T &x,T y){ x = min((x), (y));}

#ifdef LOCAL

#define fr freopen("in.txt","r",stdin)

#define fo freopen("output.txt","w",stdout)

#define error(x) cerr << #x << " = " << (x) <<endl

#define Error(a,b) cerr<<"( "<<#a<<" , "<<#b<<" ) = ( "<<(a)<<" , "<<(b)<<" )\n";

#define errop(a) cerr<<#a<<" = ( "<<((a).x)<<" , "<<((a).y)<<" )\n";

#define dbg(x) F(i,0,sz(x)) cout << x[i] << " ";

#define dbgg(x) F(i,0,sz(x)){F(j,0,sz(x[0])) cout << x[i][j] << " "; cout << endl;}

#else

#define fr 1

#define fo 1

#define error(x)

#define Error(a,b)

#define errop(a)

#define dbg(x)

#define dbgg(x)

#endif

inline void sc(int &x)

{

bool f=0; x=0; char c=getchar();

while((c<'0' || c>'9') && c!='-') c=getchar();

if(c=='-') { f=1; c=getchar(); }

while(c>='0' && c<='9') { x=x\*10+c-'0'; c=getchar(); }

if(f) x=-x; return;

}

void sc(int &x,int &y) { sc(x); return sc(y); }

void sc(int &x,int &y,int &z) { sc(x); sc(y); return sc(z); }

#define inf 1e9

const double eps = 1e-7;

int main(){

std::ios::sync\_with\_stdio(false);

//fr;

int i, j, k, m, n;

sc(m, n, k);

k--;

vector<char> vv = {'L', 'R'};

cout << k / (2 \* n) + 1 << " " << (k % (2 \* n)) / 2 + 1 << " " << vv[(k % (2 \* n)) % 2] << endl;

return 0;

}

<http://codeforces.com/contest/752/problem/D>

D. Santa Claus and a Palindrome

time limit per test

2 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

Santa Claus likes palindromes very much. There was his birthday recently. *k* of his friends came to him to congratulate him, and each of them presented to him a string *si* having the same length *n*. We denote the beauty of the *i*-th string by *ai*. It can happen that *ai* is negative — that means that Santa doesn't find this string beautiful at all.

Santa Claus is crazy about palindromes. He is thinking about the following question: what is the maximum possible total beauty of a palindrome which can be obtained by concatenating some (possibly all) of the strings he has? Each present can be used at most once. Note that all strings have **the same length** *n*.

Recall that a palindrome is a string that doesn't change after one reverses it.

Since the empty string is a palindrome too, the answer can't be negative. Even if all *ai*'s are negative, Santa can obtain the empty string.

**Input**

The first line contains two positive integers *k* and *n* divided by space and denoting the number of Santa friends and the length of every string they've presented, respectively (1 ≤ *k*, *n* ≤ 100 000; *n*·*k*  ≤ 100 000).

*k* lines follow. The *i*-th of them contains the string *si* and its beauty *ai* ( - 10 000 ≤ *ai* ≤ 10 000). The string consists of *n* lowercase English letters, and its beauty is integer. Some of strings may coincide. Also, equal strings can have different beauties.

**Output**

In the only line print the required maximum possible beauty.

**Examples**

**input**

7 3  
abb 2  
aaa -3  
bba -1  
zyz -4  
abb 5  
aaa 7  
xyx 4

**output**

12

**input**

3 1  
a 1  
a 2  
a 3

**output**

6

**input**

2 5  
abcde 10000  
abcde 10000

**output**

0

**Note**

In the first example Santa can obtain abbaaaxyxaaabba by concatenating strings 5, 2, 7, 6 and 3 (in this order).

#include <bits/stdc++.h>

using namespace std;

const int inf = 1e9;

int k, n;

map<string, priority\_queue<int> > mp;

vector<string> un;

int main() {

ios\_base::sync\_with\_stdio(0);

cin >> k >> n;

for (int i = 0; i < k; i++) {

string s; int val;

cin >> s >> val;

if (mp[s].empty()) {

un.push\_back(s);

mp[s].push(-inf);

}

mp[s].push(val);

}

int res = 0, mn = 0;

for (int i = 0; i < un.size(); i++) {

string a = un[i];

string b = un[i];

reverse(b.begin(), b.end());

if (a == b) {

while (mp[a].size() > 1) {

int x = mp[a].top(); mp[a].pop();

int y = mp[a].top(); mp[a].pop();

if (x >= 0 && y >= 0) {

res += x + y;

}

else if (x >= 0) {

int t = max(-x, y);

res += x + t;

mn = min(mn, t);

}

}

}

else {

while (!mp[a].empty() && !mp[b].empty()) {

int x = mp[a].top(); mp[a].pop();

int y = mp[b].top(); mp[b].pop();

if (x + y > 0) res += x + y;

}

}

}

res -= mn;

cout << res << "\n";

return 0;

}

<http://codeforces.com/contest/580/problem/A>

A. Kefa and First Steps

time limit per test

2 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

Kefa decided to make some money doing business on the Internet for exactly *n* days. He knows that on the *i*-th day (1 ≤ *i* ≤ *n*) he makes *ai* money. Kefa loves progress, that's why he wants to know the length of the maximum non-decreasing subsegment in sequence *ai*. Let us remind you that the subsegment of the sequence is its continuous fragment. A subsegment of numbers is called non-decreasing if all numbers in it follow in the non-decreasing order.

Help Kefa cope with this task!

**Input**

The first line contains integer *n* (1 ≤ *n* ≤ 105).

The second line contains *n* integers *a*1,  *a*2,  ...,  *an* (1 ≤ *ai* ≤ 109).

**Output**

Print a single integer — the length of the maximum non-decreasing subsegment of sequence *a*.

**Examples**

**input**

6  
2 2 1 3 4 1

**output**

3

**input**

3  
2 2 9

**output**

3

**Note**

In the first test the maximum non-decreasing subsegment is the numbers from the third to the fifth one.

In the second test the maximum non-decreasing subsegment is the numbers from the first to the third one.

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <iostream>

#include <memory.h>

#include <math.h>

#include <assert.h>

#include <queue>

#include <map>

#include <set>

#include <string>

#include <algorithm>

#include <functional>

#include <vector>

#include <stack>

using namespace std;

typedef long long ll;

typedef unsigned long long ull;

typedef pair<int, int> Pi;

#define Fi first

#define Se second

#define pb(x) push\_back(x)

#define sz(x) (int)x.size()

#define rep(i, n) for(int i=0;i<n;i++)

#define repp(i, n) for(int i=1;i<=n;i++)

#define all(x) x.begin(), x.end()

#define ABS(x) (((x) > 0 ) ? (x) : (-(x)))

#define MAX2(x, y) (((x) > (y)) ? (x) : (y))

#define MIN2(x, y) (((x) < (y)) ? (x) : (y))

#define MAX3(x, y, z) ( (x) > (y) ? ( (x) > (z) ? (x) : (z) ) : ( (y) > (z) ? (y) : (z) ) )

#define MIN3(x, y, z) ( (x) < (y) ? ( (x) < (z) ? (x) : (z) ) : ( (y) < (z) ? (y) : (z) ) )

#define MID3(val1,val2,val3) MAX2(MIN2(MAX2(val1,val2),val3),MIN2(val1,val2))

#define geti1(X) scanf("%d",&X)

#define geti2(X,Y) scanf("%d%d",&X,&Y)

#define geti3(X,Y,Z) scanf("%d%d%d",&X,&Y,&Z)

#define geti4(X,Y,Z,W) scanf("%d%d%d%d",&X,&Y,&Z,&W)

#define GET\_MACRO(\_1,\_2,\_3,\_4,NAME,...) NAME

#define geti(...) GET\_MACRO(\_\_VA\_ARGS\_\_, geti4, geti3, geti2, geti1) (\_\_VA\_ARGS\_\_)

#define INF 987654321

#define IINF 987654321987654321

int N,M,K;

int main(void)

{

geti(N);

int cur = -1; int len = 0; int ans = 0;

repp(i,N){

int x; geti(x);

if( x >= cur ){

len++; cur =x;

}else{

len = 1; cur = x;

}

ans = max(len,ans);

}

cout << ans;

}

<http://codeforces.com/contest/102/problem/B>

B. Sum of Digits

time limit per test

2 seconds

memory limit per test

265 megabytes

input

standard input

output

standard output

Having watched the last Harry Potter film, little Gerald also decided to practice magic. He found in his father's magical book a spell that turns any number in the sum of its digits. At the moment Gerald learned that, he came across a number *n*. How many times can Gerald put a spell on it until the number becomes one-digit?

**Input**

The first line contains the only integer *n* (0 ≤ *n* ≤ 10100000). It is guaranteed that *n* doesn't contain any leading zeroes.

**Output**

Print the number of times a number can be replaced by the sum of its digits until it only contains one digit.

**Examples**

**input**

0

**output**

0

**input**

10

**output**

1

**input**

991

**output**

3

**Note**

In the first sample the number already is one-digit — Herald can't cast a spell.

The second test contains number 10. After one casting of a spell it becomes 1, and here the process is completed. Thus, Gerald can only cast the spell once.

The third test contains number 991. As one casts a spell the following transformations take place: 991 → 19 → 10 → 1. After three transformations the number becomes one-digit.

#include <algorithm>

#include <assert.h>

#include <bitset>

#include <complex>

#include <ctime>

#include <fstream>

#include <iomanip>

#include <limits.h>

#include <list>

#include <map>

#include <math.h>

#include <queue>

#include <set>

#include <stack>

#include <stdio.h>

#include <string>

#include <unordered\_map>

#include <unordered\_set>

#include <utility>

#include <vector>

using namespace std;

typedef long long LL;

typedef vector<int> VI;

typedef vector<VI> VVI;

typedef vector<LL> VLL;

typedef pair<LL, LL> PLL;

typedef pair<int,int> PII;

typedef pair<int, int> PII;

typedef pair<LL, PLL> PLPLL;

typedef pair<PLL,LL> PPLLL;

typedef pair<LL, char> PLC;

typedef pair<char, char> PCC;

typedef pair<LL, VLL> PVLL;

typedef vector<PLL> VPLL;

typedef vector<PII> VPII;

typedef vector<std::string> VS;

typedef std::vector<LL>::iterator VLLitr;

#define arjun int main()

#define FOR(i, x, y) for (LL i = (x); i < (y); ++i)

#define FORC(i, x, y, in) for (LL i = (x); i < (y); i = i + in)

#define RFOR(i, x, y) for (LL i = (x); i >= (y); i--)

#define FORO(it, c) \

for (\_\_typeof((c).begin()) it = (c).begin(); it != (c).end(); it++)

#define pb(e) push\_back(e)

#define mp make\_pair

#define F first

#define S second

#define All(x) x.begin(), x.end()

#define fast \

ios\_base::sync\_with\_stdio(false); \

cin.tie(NULL);

#define PI 3.14159265

#define RESET(a, b) memset(a, b, sizeof(a))

template <typename T> T maxm(T a, T b) { return (a > b) ? a : b; }

template <typename T> T minm(T a, T b) { return (a < b) ? a : b; }

template <typename T> T power(T e, T n) {

T x = 1, p = e;

while (n) {

if (n & 1)

x = x \* p;

p = p \* p;

n >>= 1;

}

return x;

}

template <typename T> T powerm(T e, T n, T m) {

T x = 1, p = e;

while (n) {

if (n & 1)

x = (x \* p) % m;

p = (p \* p) % m;

n >>= 1;

}

return x;

}

template <typename T> T InverseEuler(T a, T m) {

return (a == 1 ? 1 : power(a, m - 2, m));

}

template <typename T> T lcm(T a, T b) { return (a \* (b / \_\_gcd(a, b))); }

string StringToUpper(string strToConvert) {

std::transform(strToConvert.begin(), strToConvert.end(), strToConvert.begin(),

::toupper);

return strToConvert;

}

string StringToLower(string strToConvert) {

std::transform(strToConvert.begin(), strToConvert.end(), strToConvert.begin(),

::tolower);

return strToConvert;

}

LL MAX = (LL)(3 \* 1e18);

LL MOD = (LL)(1e9 + 7);

#ifdef JUDGE

#include <fstream>

std::ifstream cin("input.txt");

std::ofstream cout("output.txt");

#else

#include <iostream>

using std::cin;

using std::cout;

#endif

string func(string s){

LL sum = 0;

FOR(i,0,s.length()){

sum+=(s[i]-'0');

}

return to\_string(sum);

}

arjun {

fast;

string s;

cin>>s;

LL ans = 0;

while((s).length()>1){

ans++;

(s = func(s));

}

cout<<ans<<endl;

}

<http://codeforces.com/contest/385/problem/A>

A. Bear and Raspberry

time limit per test

1 second

memory limit per test

256 megabytes

input

standard input

output

standard output

The bear decided to store some raspberry for the winter. He cunningly found out the price for a barrel of honey in kilos of raspberry for each of the following *n* days. According to the bear's data, on the *i*-th (1 ≤ *i* ≤ *n*) day, the price for one barrel of honey is going to is *xi*kilos of raspberry.

Unfortunately, the bear has neither a honey barrel, nor the raspberry. At the same time, the bear's got a friend who is ready to lend him a barrel of honey for exactly one day for *c* kilograms of raspberry. That's why the bear came up with a smart plan. He wants to choose some day *d* (1 ≤ *d* < *n*), lent a barrel of honey and immediately (on day *d*) sell it according to a daily exchange rate. The next day (*d* + 1) the bear wants to buy a new barrel of honey according to a daily exchange rate (as he's got some raspberry left from selling the previous barrel) and immediately (on day *d* + 1) give his friend the borrowed barrel of honey as well as *c* kilograms of raspberry for renting the barrel.

The bear wants to execute his plan at most once and then hibernate. What maximum number of kilograms of raspberry can he earn? Note that if at some point of the plan the bear runs out of the raspberry, then he won't execute such a plan.

**Input**

The first line contains two space-separated integers, *n* and *c* (2 ≤ *n* ≤ 100, 0 ≤ *c* ≤ 100), — the number of days and the number of kilos of raspberry that the bear should give for borrowing the barrel.

The second line contains *n* space-separated integers *x*1, *x*2, ..., *xn* (0 ≤ *xi* ≤ 100), the price of a honey barrel on day *i*.

**Output**

Print a single integer — the answer to the problem.

**Examples**

**input**

5 1  
5 10 7 3 20

**output**

3

**input**

6 2  
100 1 10 40 10 40

**output**

97

**input**

3 0  
1 2 3

**output**

0

**Note**

In the first sample the bear will lend a honey barrel at day 3 and then sell it for 7. Then the bear will buy a barrel for 3 and return it to the friend. So, the profit is (7 - 3 - 1) = 3.

In the second sample bear will lend a honey barrel at day 1 and then sell it for 100. Then the bear buy the barrel for 1 at the day 2. So, the profit is (100 - 1 - 2) = 97.

#include<stdio.h>

main ()

{

int d, k, i, j, x, sum=0;

scanf("%d %d", &d, &k);

int a[d];

for(i=0; i<d; i++)

{

scanf("%d", &a[i]);

}

for(i=0; i<d-1; i++)

{

if(a[i]<a[i+1]) continue;

else

{

x=a[i]-a[i+1];

if(x>sum) sum=x;

else sum=sum;

}

}

if(sum==0) printf("0");

else

{

if(sum>k) printf("%d", (sum-k));

else printf("0");

}

return 0;

}

<http://codeforces.com/contest/116/problem/C>

C. Party

time limit per test

3 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

A company has *n* employees numbered from 1 to *n*. Each employee either has no immediate manager or exactly one immediate manager, who is another employee with a different number. An employee *A* is said to be the *superior* of another employee *B* if at least one of the following is true:

* Employee *A* is the immediate manager of employee *B*
* Employee *B* has an immediate manager employee *C* such that employee *A* is the superior of employee *C*.

The company will not have a managerial cycle. That is, there will not exist an employee who is the superior of his/her own immediate manager.

Today the company is going to arrange a party. This involves dividing all *n* employees into several groups: every employee must belong to exactly one group. Furthermore, within any single group, there must not be two employees *A* and *B* such that *A* is the superior of *B*.

What is the minimum number of groups that must be formed?

**Input**

The first line contains integer *n* (1 ≤ *n* ≤ 2000) — the number of employees.

The next *n* lines contain the integers *pi* (1 ≤ *pi* ≤ *n* or *pi* = -1). Every *pi* denotes the immediate manager for the *i*-th employee. If *pi* is -1, that means that the *i*-th employee does not have an immediate manager.

It is guaranteed, that no employee will be the immediate manager of him/herself (*pi* ≠ *i*). Also, there will be no managerial cycles.

**Output**

Print a single integer denoting the minimum number of groups that will be formed in the party.

**Examples**

**input**

5  
-1  
1  
2  
1  
-1

**output**

3

**Note**

For the first example, three groups are sufficient, for example:

* Employee 1
* Employees 2 and 4
* Employees 3 and 5

#include <bits/stdc++.h>

using namespace std;

typedef long long ll;

typedef vector<int> vi;

typedef pair<int, int> pii;

typedef set<int> si;

typedef map<int,int> mii;

#define FOR(i, a, b) for (int i=a; i<b; i++)

#define F0R(i, a) for (int i=0; i<a; i++)

#define FORd(i,a,b) for (int i = b-1; i >= a; i--)

#define F0Rd(i,a) for (int i = a-1; i >= 0; i--)

#define mp make\_pair

#define pb push\_back

#define f first

#define s second

#define lb lower\_bound

#define ub upper\_bound

const int MOD = 1000000007;

double PI = 4\*atan(1);

int maxh = 0, par[2001], height[2001];

int geth(int node) {

if (height[node] != 0) return height[node];

if (par[node] == -1) return height[node] = 1;

return height[node]=geth(par[node])+1;

}

int main() {

ios\_base::sync\_with\_stdio(0);cin.tie(0);

int n; cin >> n;

FOR(i,1,n+1) cin >> par[i];

FOR(i,1,n+1) maxh = max(maxh,geth(i));

cout << maxh;

}

<http://codeforces.com/contest/385/problem/B>

B. Bear and Strings

time limit per test

1 second

memory limit per test

256 megabytes

input

standard input

output

standard output

The bear has a string *s* = *s*1*s*2... *s*|*s*| (record |*s*| is the string's length), consisting of lowercase English letters. The bear wants to count the number of such pairs of indices *i*, *j* (1 ≤ *i* ≤ *j* ≤ |*s*|), that string *x*(*i*, *j*) = *sisi*+ 1... *sj* contains at least one string "bear" as a substring.

String *x*(*i*, *j*) contains string "bear", if there is such index *k* (*i* ≤ *k* ≤ *j* - 3), that *sk* = *b*, *sk*+ 1 = *e*, *sk*+ 2 = *a*, *sk*+ 3 = *r*.

Help the bear cope with the given problem.

**Input**

The first line contains a non-empty string *s* (1 ≤ |*s*| ≤ 5000). It is guaranteed that the string only consists of lowercase English letters.

**Output**

Print a single number — the answer to the problem.

**Examples**

**input**

bearbtear

**output**

6

**input**

bearaabearc

**output**

20

**Note**

In the first sample, the following pairs (*i*, *j*) match: (1, 4), (1, 5), (1, 6), (1, 7), (1, 8), (1, 9).

In the second sample, the following pairs (*i*, *j*) match: (1,  4), (1,  5), (1,  6), (1,  7), (1,  8), (1,  9), (1,  10), (1,  11), (2,  10), (2,  11), (3,  10), (3,  11), (4,  10), (4,  11), (5,  10), (5,  11), (6,  10), (6,  11), (7,  10), (7,  11).

#include <bits/stdc++.h>

#define pii pair <int,int>

#define pll pair <long long,long long>

#define sc scanf

#define pf printf

#define Pi 2\*acos(0.0)

#define ms(a,b) memset(a, b, sizeof(a))

#define pb(a) push\_back(a)

#define MP make\_pair

#define db double

#define ll long long

#define EPS 10E-10

#define ff first

#define ss second

#define sqr(x) (x)\*(x)

#define D(x) cout<<#x " = "<<(x)<<endl

#define VI vector <int>

#define DBG pf("Hi\n")

#define MOD 1000000007

#define CIN ios\_base::sync\_with\_stdio(0); cin.tie(0); cout.tie(0)

#define SZ(a) (int)a.size()

#define sf(a) scanf("%d",&a)

#define sfl(a) scanf("%lld",&a)

#define sff(a,b) scanf("%d %d",&a,&b)

#define sffl(a,b) scanf("%lld %lld",&a,&b)

#define sfff(a,b,c) scanf("%d %d %d",&a,&b,&c)

#define sfffl(a,b,c) scanf("%lld %lld %lld",&a,&b,&c)

#define stlloop(v) for(\_\_typeof(v.begin()) it=v.begin();it!=v.end();it++)

#define loop(i,n) for(int i=0;i<n;i++)

#define loop1(i,n) for(int i=1;i<=n;i++)

#define REP(i,a,b) for(int i=a;i<b;i++)

#define RREP(i,a,b) for(int i=a;i>=b;i--)

#define TEST\_CASE(t) for(int z=1;z<=t;z++)

#define PRINT\_CASE printf("Case %d: ",z)

#define LINE\_PRINT\_CASE printf("Case %d:\n",z)

#define CASE\_PRINT cout<<"Case "<<z<<": "

#define all(a) a.begin(),a.end()

#define intlim 2147483648

#define infinity (1<<28)

#define ull unsigned long long

#define gcd(a, b) \_\_gcd(a, b)

#define lcm(a, b) ((a)\*((b)/gcd(a,b)))

using namespace std;

/\*----------------------Graph Moves----------------\*/

//const int fx[]={+1,-1,+0,+0};

//const int fy[]={+0,+0,+1,-1};

//const int fx[]={+0,+0,+1,-1,-1,+1,-1,+1}; // Kings Move

//const int fy[]={-1,+1,+0,+0,+1,+1,-1,-1}; // Kings Move

//const int fx[]={-2, -2, -1, -1, 1, 1, 2, 2}; // Knights Move

//const int fy[]={-1, 1, -2, 2, -2, 2, -1, 1}; // Knights Move

/\*------------------------------------------------\*/

/\*-----------------------Bitmask------------------\*/

//int Set(int N,int pos){return N=N | (1<<pos);}

//int reset(int N,int pos){return N= N & ~(1<<pos);}

//bool check(int N,int pos){return (bool)(N & (1<<pos));}

/\*------------------------------------------------\*/

string str;

int cum[505];

vector<int>v;

int main()

{

///freopen("in.txt","r",stdin);

///freopen("out.txt","w",stdout);

cin>>str;

int ans=0;

bool test=0;

int l=SZ(str);

int cnt=0;

for(int i=0;i+3<SZ(str);i++)

{

if(str[i]=='b')

{

string temp=str.substr(i,4);

if(temp=="bear")

{

int n=l-(i+3);

if(i==0)

ans+=(n);

else if(n==1)

ans+=i+1;

else

ans+=(n\*(i+1));

ans-=cnt\*n;

cnt=i+1;

}

}

}

cout<<ans<<endl;

return 0;

}

<http://codeforces.com/contest/118/problem/A>

A. String Task

time limit per test

2 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

Petya started to attend programming lessons. On the first lesson his task was to write a simple program. The program was supposed to do the following: in the given string, consisting if uppercase and lowercase Latin letters, it:

* deletes all the vowels,
* inserts a character "." before each consonant,
* replaces all uppercase consonants with corresponding lowercase ones.

Vowels are letters "A", "O", "Y", "E", "U", "I", and the rest are consonants. The program's input is exactly one string, it should return the output as a single string, resulting after the program's processing the initial string.

Help Petya cope with this easy task.

**Input**

The first line represents input string of Petya's program. This string only consists of uppercase and lowercase Latin letters and its length is from 1 to 100, inclusive.

**Output**

Print the resulting string. It is guaranteed that this string is not empty.

**Examples**

**input**

tour

**output**

.t.r

**input**

Codeforces

**output**

.c.d.f.r.c.s

**input**

aBAcAba

**output**

.b.c.b

#include <iostream>

#include <vector>

#include <string>

using namespace std;

int main(void)

{

vector<char> s;

string str;

const string aer{"ayeiou"};

getline(cin,str);

for(int i=0; i<str.size() ;++i)

str[i] = str[i]<91?str[i]+32:str[i];

for(int i=0; i<str.size() ;++i)

{

string tt{"2"};

tt[0] = str[i];

if(aer.find(tt) == string::npos)

s.push\_back(str[i]);

}

for(auto i:s)

cout << '.' << i;

return 0;

}

<http://codeforces.com/contest/4/problem/A>

A. Watermelon

time limit per test

1 second

memory limit per test

64 megabytes

input

standard input

output

standard output

One hot summer day Pete and his friend Billy decided to buy a watermelon. They chose the biggest and the ripest one, in their opinion. After that the watermelon was weighed, and the scales showed *w* kilos. They rushed home, dying of thirst, and decided to divide the berry, however they faced a hard problem.

Pete and Billy are great fans of even numbers, that's why they want to divide the watermelon in such a way that each of the two parts weighs even number of kilos, at the same time it is not obligatory that the parts are equal. The boys are extremely tired and want to start their meal as soon as possible, that's why you should help them and find out, if they can divide the watermelon in the way they want. For sure, each of them should get a part of positive weight.

**Input**

The first (and the only) input line contains integer number *w* (1 ≤ *w* ≤ 100) — the weight of the watermelon bought by the boys.

**Output**

Print YES, if the boys can divide the watermelon into two parts, each of them weighing even number of kilos; and NO in the opposite case.

**Examples**

**input**

8

**output**

YES

**Note**

For example, the boys can divide the watermelon into two parts of 2 and 6 kilos respectively (another variant — two parts of 4 and 4 kilos).

#include <iostream>

using namespace std;

int main()

{

int w; cin >> w;

cout << ((w % 2 == 0 && w != 2) ? "YES\n" : "NO\n");

return 0;

}

<http://codeforces.com/contest/520/problem/A>

A. Pangram

time limit per test

2 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

A word or a sentence in some language is called a *pangram* if all the characters of the alphabet of this language appear in it *at least once*. Pangrams are often used to demonstrate fonts in printing or test the output devices.

You are given a string consisting of lowercase and uppercase Latin letters. Check whether this string is a pangram. We say that the string contains a letter of the Latin alphabet if this letter occurs in the string in uppercase or lowercase.

**Input**

The first line contains a single integer *n* (1 ≤ *n* ≤ 100) — the number of characters in the string.

The second line contains the string. The string consists only of uppercase and lowercase Latin letters.

**Output**

Output "YES", if the string is a pangram and "NO" otherwise.

**Examples**

**input**

12  
toosmallword

**output**

NO

**input**

35  
TheQuickBrownFoxJumpsOverTheLazyDog

**output**

YES

#include<iostream>

#include<stdio.h>

#include<math.h>

#include<string>

#include<map>

#define ll long long

using namespace std;

int main()

{

//freopen("input.txt", "r", stdin);

//freopen("output.txt", "w", stdout);

int n;

cin >> n;

string a;

map<char, bool> w;

int col = 0;

cin >> a;

for(auto i: a)

{

if(w[tolower(i)] == 0)

{

col++;

w[tolower(i)] = 1;

}

}

if(col == 26)

cout << "YES";

else

cout << "NO";

return 0;

}

<http://codeforces.com/contest/747/problem/C>

C. Servers

time limit per test

2 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

There are *n* servers in a laboratory, each of them can perform tasks. Each server has a unique id — integer from 1 to *n*.

It is known that during the day *q* tasks will come, the *i*-th of them is characterized with three integers: *ti* — the moment in seconds in which the task will come, *ki* — the number of servers needed to perform it, and *di* — the time needed to perform this task in seconds. All *ti* are distinct.

To perform the *i*-th task you need *ki* servers which are unoccupied in the second *ti*. After the servers begin to perform the task, each of them will be busy over the next *di* seconds. Thus, they will be busy in seconds *ti*, *ti* + 1, ..., *ti* + *di* - 1. For performing the task, *ki*servers with the smallest ids will be chosen from all the unoccupied servers. If in the second *ti* there are not enough unoccupied servers, the task is ignored.

Write the program that determines which tasks will be performed and which will be ignored.

**Input**

The first line contains two positive integers *n* and *q* (1 ≤ *n* ≤ 100, 1 ≤ *q* ≤ 105) — the number of servers and the number of tasks.

Next *q* lines contains three integers each, the *i*-th line contains integers *ti*, *ki* and *di* (1 ≤ *ti* ≤ 106, 1 ≤ *ki* ≤ *n*, 1 ≤ *di* ≤ 1000) — the moment in seconds in which the *i*-th task will come, the number of servers needed to perform it, and the time needed to perform this task in seconds. The tasks are given in a chronological order and they will come in distinct seconds.

**Output**

Print *q* lines. If the *i*-th task will be performed by the servers, print in the *i*-th line the sum of servers' ids on which this task will be performed. Otherwise, print -1.

**Examples**

**input**

4 3  
1 3 2  
2 2 1  
3 4 3

**output**

6  
-1  
10

**input**

3 2  
3 2 3  
5 1 2

**output**

3  
3

**input**

8 6  
1 3 20  
4 2 1  
6 5 5  
10 1 1  
15 3 6  
21 8 8

**output**

6  
9  
30  
-1  
15  
36

**Note**

In the first example in the second 1 the first task will come, it will be performed on the servers with ids 1, 2 and 3 (the sum of the ids equals 6) during two seconds. In the second 2 the second task will come, it will be ignored, because only the server 4 will be unoccupied at that second. In the second 3 the third task will come. By this time, servers with the ids 1, 2 and 3 will be unoccupied again, so the third task will be done on all the servers with the ids 1, 2, 3 and 4 (the sum of the ids is 10).

In the second example in the second 3 the first task will come, it will be performed on the servers with ids 1 and 2 (the sum of the ids is 3) during three seconds. In the second 5 the second task will come, it will be performed on the server 3, because the first two servers will be busy performing the first task.

#include <bits/stdc++.h>

using namespace std;

#define ll long long int

const int N=2e5+5;

const int mod=1e9;

bool cmp( pair< int,pair<int,int> >p1,pair< int,pair<int,int> >p2)

{

if(p1.first < p2.first)

return true;

else

return false;

}

int main()

{

int n,q;

cin>>n>>q;

int a[n+1];

int b[n+1];

for(int i=0;i<=n;i++) a[i]=0,b[i]=0;

vector< pair<int,pair<int,int> > >v;

v.clear();

int f=q;

while(q--){

int x,y,z;

cin>>x>>y>>z;

v.push\_back(make\_pair(x,make\_pair(y,z-1)));

}

int available=n;

// sort(v.begin(),v.end(),cmp);

int len=v.size();

int lst=0;

int ans=0;

for(int i=0;i<f;i++)

{

if(i==0)

{

if(v[i].second.first <= available)

{

for(int j=1;j<=v[i].second.first;j++)

{

b[j]=1;

a[j]=v[i].first+v[i].second.second;

ans+=j;

available--;

}

cout<<ans<<endl;

}

else

cout<<"-1\n";

continue;

}

for(int k=1;k<=n;k++)

{

if(a[k] < v[i].first && b[k])

{

a[k]=0;

b[k]=0;

available++;

}

}

if(v[i].second.first > available)

{

cout<<"-1\n";

continue;

}

else

{

ans=0;

int tot=0;

for(int k=1;k<=n;k++)

{

if(!b[k] && a[k]<v[i].second.second +v[i].first)

{

b[k]=1;

a[k]=v[i].first+v[i].second.second;

ans+=k;

tot++;

available--;

}

if(tot >= v[i].second.first)

break;

}

cout<<ans<<endl;

}

}

return 0;

}

<http://codeforces.com/contest/550/problem/B>

B. Preparing Olympiad

time limit per test

2 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

You have *n* problems. You have estimated the difficulty of the *i*-th one as integer *ci*. Now you want to prepare a problemset for a contest, using some of the problems you've made.

A problemset for the contest must consist of at least two problems. You think that the total difficulty of the problems of the contest must be at least *l* and at most *r*. Also, you think that the difference between difficulties of the easiest and the hardest of the chosen problems must be at least *x*.

Find the number of ways to choose a problemset for the contest.

**Input**

The first line contains four integers *n*, *l*, *r*, *x* (1 ≤ *n* ≤ 15, 1 ≤ *l* ≤ *r* ≤ 109, 1 ≤ *x* ≤ 106) — the number of problems you have, the minimum and maximum value of total difficulty of the problemset and the minimum difference in difficulty between the hardest problem in the pack and the easiest one, respectively.

The second line contains *n* integers *c*1, *c*2, ..., *cn* (1 ≤ *ci* ≤ 106) — the difficulty of each problem.

**Output**

Print the number of ways to choose a suitable problemset for the contest.

**Examples**

**input**

3 5 6 1  
1 2 3

**output**

2

**input**

4 40 50 10  
10 20 30 25

**output**

2

**input**

5 25 35 10  
10 10 20 10 20

**output**

6

**Note**

In the first example two sets are suitable, one consisting of the second and third problem, another one consisting of all three problems.

In the second example, two sets of problems are suitable — the set of problems with difficulties 10 and 30 as well as the set of problems with difficulties 20 and 30.

In the third example any set consisting of one problem of difficulty 10 and one problem of difficulty 20 is suitable.

#include <bits/stdc++.h>

using namespace std;

int n;

long a[20],mi,ma,dif,minum=9000000000,manum=0,fin=0;

long long x=0;

int main()

{

ios::sync\_with\_stdio(false);

cin.tie(0);

cout.tie(0);

cin >>n>>mi>>ma>>dif;

for (int i=0;i<n;i++)

cin>> a[i];

for (int i=0;i<pow(2,n);i++)

{

for (int j=0;j<n;j++)

{

if ((1<<j)&i)

{

x+=a[j];

if (a[j]<minum)

minum=a[j];

if (a[j]>manum)

manum=a[j];

}

}

if (x>=mi&&x<=ma&&manum-minum>=dif)

fin++;

x=0;

manum=0;

minum=9000000000;

}

cout <<fin;

return 0;

}

<http://codeforces.com/contest/21/problem/C>

C. Stripe 2

time limit per test

1 second

memory limit per test

64 megabytes

input

standard input

output

standard output

Once Bob took a paper stripe of n squares (the height of the stripe is 1 square). In each square he wrote an integer number, possibly negative. He became interested in how many ways exist to cut this stripe into three pieces so that the sum of numbers from each piece is equal to the sum of numbers from any other piece, and each piece contains positive integer amount of squares. Would you help Bob solve this problem?

**Input**

The first input line contains integer *n* (1 ≤ *n* ≤ 105) — amount of squares in the stripe. The second line contains n space-separated numbers — they are the numbers written in the squares of the stripe. These numbers are integer and do not exceed 10000 in absolute value.

**Output**

Output the amount of ways to cut the stripe into three non-empty pieces so that the sum of numbers from each piece is equal to the sum of numbers from any other piece. Don't forget that it's allowed to cut the stripe along the squares' borders only.

**Examples**

**input**

4  
1 2 3 3

**output**

1

**input**

5  
1 2 3 4 5

**output**

0

#include <bits/stdc++.h>

#define vi vector <int>

#define vvi vector < vi >

#define ii pair < int,int >

#define vii vector < ii >

#define all(v) ((v).begin()), ((v).end())

#define sz(v) ((ll)((v).size()))

#define clr(v, d) memset(v, d, sizeof(v))

#define rep(i, v) for(ll i=0;i<sz(v);++i)

#define lp(i, j, n) for(ll i=(j);i<(ll)(n);++i)

#define lpd(i, j, n) for(ll i=(j);i>=(ll)(n);--i)

#define pb push\_back

#define MP make\_pair

#define READ freopen("in.txt","r",stdin)

#define WRITE freopen("output.txt","w",stdout)

#define scf(n) scanf("%d",&n)

#define FastIO ios\_base::sync\_with\_stdio(false)

#define vb vector <bool>

#define ull unsigned long long

typedef long long ll;

const ll oo = 1e9;

const ll mod = 95542721;

const ll MAX =1e9;

const ll ARR =410;

using namespace std;

int main() {

#ifndef ONLINE\_JUDGE

READ;

#endif

FastIO;

// clr(cnt,0);

int n;

cin>>n;

int a[n],cnt[n];

clr(cnt,0);

ll s=0,res=0;

ll sum=0;

lp(i,0,n){

cin>>a[i];

sum+=a[i];

}

if(sum%3!=0)return cout<<0,0;

sum/=3;

for(int i=n-1;i>=0;i--){

s+=a[i];

if(sum==s){

cnt[i]=1;

}

}

for(int i = n-2 ; i >= 0 ; --i)

cnt[i] += cnt[i+1];

s=0;

lp(i,0,n-2){

s+=a[i];

if(s==sum)

res+=cnt[i+2];

}

cout<<res;

return 0;

}

<http://codeforces.com/contest/114/problem/C>

C. Grammar Lessons

time limit per test

5 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

Petya got interested in grammar on his third year in school. He invented his own language called Petya's. Petya wanted to create a maximally simple language that would be enough to chat with friends, that's why all the language's grammar can be described with the following set of rules:

* There are three parts of speech: the adjective, the noun, the verb. Each word in his language is an adjective, noun or verb.
* There are two genders: masculine and feminine. Each word in his language has gender either masculine or feminine.
* Masculine adjectives end with -lios, and feminine adjectives end with -liala.
* Masculine nouns end with -etr, and feminime nouns end with -etra.
* Masculine verbs end with -initis, and feminime verbs end with -inites.
* Thus, each word in the Petya's language has one of the six endings, given above. There are no other endings in Petya's language.
* It is accepted that the whole word consists of an ending. That is, words "lios", "liala", "etr" and so on belong to the Petya's language.
* There aren't any punctuation marks, grammatical tenses, singular/plural forms or other language complications.
* A sentence is either exactly one valid language word or exactly one *statement*.

*Statement* is any sequence of the Petya's language, that satisfy both conditions:

* Words in statement follow in the following order (from the left to the right): zero or more adjectives followed by exactly one noun followed by zero or more verbs.
* All words in the statement should have the same gender.

After Petya's friend Vasya wrote instant messenger (an instant messaging program) that supported the Petya's language, Petya wanted to add spelling and grammar checking to the program. As Vasya was in the country and Petya didn't feel like waiting, he asked you to help him with this problem. Your task is to define by a given sequence of words, whether it is true that the given text represents exactly one sentence in Petya's language.

**Input**

The first line contains one or more words consisting of lowercase Latin letters. The overall number of characters (including letters and spaces) does not exceed 105.

It is guaranteed that any two consecutive words are separated by exactly one space and the input data do not contain any other spaces. It is possible that given words do not belong to the Petya's language.

**Output**

If some word of the given text does not belong to the Petya's language or if the text contains more that one sentence, print "NO" (without the quotes). Otherwise, print "YES" (without the quotes).

**Examples**

**input**

petr

**output**

YES

**input**

etis atis animatis etis atis amatis

**output**

NO

**input**

nataliala kataliala vetra feinites

**output**

YES

#include <bits/stdc++.h>

using namespace std;

typedef long long ll;

typedef vector<int> vi;

typedef pair<int, int> pii;

typedef set<int> si;

typedef map<int,int> mii;

#define FOR(i, a, b) for (int i=a; i<b; i++)

#define F0R(i, a) for (int i=0; i<a; i++)

#define FORd(i,a,b) for (int i = b-1; i >= a; i--)

#define F0Rd(i,a) for (int i = a-1; i >= 0; i--)

#define mp make\_pair

#define pb push\_back

#define f first

#define s second

#define lb lower\_bound

#define ub upper\_bound

const int MOD = 1000000007;

double PI = 4\*atan(1);

vi gen, type;

int main() {

ios\_base::sync\_with\_stdio(0);cin.tie(0);

string s;

while (cin >> s) {

reverse(s.begin(),s.end());

if (s.length()>=4 && s.substr(0,4) == "soil") gen.pb(0), type.pb(0);

else if (s.length()>=5 && s.substr(0,5) == "alail") gen.pb(1), type.pb(0);

else if (s.length()>=3 && s.substr(0,3) == "rte") gen.pb(0), type.pb(1);

else if (s.length()>=4 && s.substr(0,4) == "arte") gen.pb(1), type.pb(1);

else if (s.length()>=6 && s.substr(0,6) == "sitini") gen.pb(0), type.pb(2);

else if (s.length()>=6 && s.substr(0,6) == "setini") gen.pb(1), type.pb(2);

else {

cout << "NO";

return 0;

}

}

if (gen.size() == 1) {

cout << "YES";

return 0;

}

F0R(i,gen.size()-1) if (gen[i] != gen[i+1]) {

cout << "NO";

return 0;

}

F0R(i,type.size()-1) if (type[i+1]-type[i]>1 || type[i+1]-type[i]<0) {

cout << "NO";

return 0;

}

int ccount = 0;

F0R(i,type.size()) if (type[i] == 1) ccount++;

if (ccount != 1) {

cout << "NO";

return 0;

}

cout << "YES";

}

<http://codeforces.com/contest/723/problem/D>

D. Lakes in Berland

time limit per test

2 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

The map of Berland is a rectangle of the size *n* × *m*, which consists of cells of size 1 × 1. Each cell is either land or water. The map is surrounded by the ocean.

*Lakes* are the maximal regions of water cells, connected by sides, which are not connected with the ocean. Formally, lake is a set of water cells, such that it's possible to get from any cell of the set to any other without leaving the set and moving only to cells adjacent by the side, none of them is located on the border of the rectangle, and it's impossible to add one more water cell to the set such that it will be connected with any other cell.

You task is to fill up with the earth the minimum number of water cells so that there will be **exactly** *k* lakes in Berland. Note that the initial number of lakes on the map is **not less** than *k*.

**Input**

The first line of the input contains three integers *n*, *m* and *k* (1 ≤ *n*, *m* ≤ 50, 0 ≤ *k* ≤ 50) — the sizes of the map and the number of lakes which should be left on the map.

The next *n* lines contain *m* characters each — the description of the map. Each of the characters is either '.' (it means that the corresponding cell is water) or '\*' (it means that the corresponding cell is land).

It is guaranteed that the map contain at least *k* lakes.

**Output**

In the first line print the minimum number of cells which should be transformed from water to land.

In the next *n* lines print *m* symbols — the map after the changes. The format must strictly follow the format of the map in the input data (there is no need to print the size of the map). If there are several answers, print any of them.

It is guaranteed that the answer exists on the given data.

**Examples**

**input**

5 4 1  
\*\*\*\*  
\*..\*  
\*\*\*\*  
\*\*.\*  
..\*\*

**output**

1  
\*\*\*\*  
\*..\*  
\*\*\*\*  
\*\*\*\*  
..\*\*

**input**

3 3 0  
\*\*\*  
\*.\*  
\*\*\*

**output**

1  
\*\*\*  
\*\*\*  
\*\*\*

**Note**

In the first example there are only two lakes — the first consists of the cells (2, 2) and (2, 3), the second consists of the cell (4, 3). It is profitable to cover the second lake because it is smaller. Pay attention that the area of water in the lower left corner is not a lake because this area share a border with the ocean.

#include <bits/stdc++.h>

using namespace std;

const int SOLID = -2;

const int OPEN = -1;

vector<int> a;

set<int> ocean, banned;

int n, m, k, next\_comp, c, ans;

vector<pair<int,int> > comps;

vector<vector<int> > g;

void dfs(int v, int comp = -1, int p = 1) {

if (comp == -1) {

comp = next\_comp++;

comps.push\_back(make\_pair(0, comp));

}

a[v] = comp;

comps[comp].first++;

for (int i = 0; i < (int)g[v].size(); ++i) {

if (a[g[v][i]] == OPEN){

dfs(g[v][i], comp, p + 1);

}

}

}

int main()

{

scanf("%d %d %d", &n, &m, &k);

for (int i = 0; i < n; ++i) {

char buf[51];

scanf("%s", buf);

for (int j = 0; j < m; ++j) {

g.push\_back(vector<int>());

if (buf[j] == '\*') a.push\_back(SOLID);

else a.push\_back(OPEN);

}

}

for (int i = 0; i < n \* m; ++i) {

if (a[i] == OPEN) {

if (i - m >= 0 && a[i - m] == OPEN) g[i].push\_back(i - m);

if (i % m > 0 && a[i - 1] == OPEN) g[i].push\_back(i - 1);

if (i % m < m - 1 && a[i + 1] == OPEN) g[i].push\_back(i + 1);

if (i + m < n \* m && a[i + m] == OPEN) g[i].push\_back(i + m);

}

}

for (int i = 0; i < n \* m; ++i){

if (a[i] == OPEN) dfs(i);

}

c = next\_comp;

for (int i = 0; i < m; ++i) {

if (a[i] != SOLID)

ocean.insert(a[i]);

}

for (int i = n \* m - m; i < n \* m; ++i) {

if (a[i] != SOLID)

ocean.insert(a[i]);

}

for (int i = 0; i < n \* m; i += m) {

if (a[i] != SOLID)

ocean.insert(a[i]);

}

for (int i = m - 1; i < n \* m; i += m) {

if (a[i] != SOLID)

ocean.insert(a[i]);

}

c -= ocean.size();

sort(comps.begin(), comps.end());

int i = 0;

while (c > k) {

if (ocean.find(comps[i].second) == ocean.end()) {

banned.insert(comps[i].second);

ans += comps[i].first;

--c;

}

++i;

}

printf("%d\n", ans);

for (int i = 0; i < n; ++i) {

for (int j = 0; j < m; ++j) {

if (a[i \* m + j] == SOLID || (a[i \* m + j] != SOLID && banned.find(a[i \* m + j]) != banned.end()))

printf("\*");

else printf(".");

}

printf("\n");

}

return 0;

}

<http://codeforces.com/contest/742/problem/E>

E. Arpa’s overnight party and Mehrdad’s silent entering

time limit per test

1 second

memory limit per test

256 megabytes

input

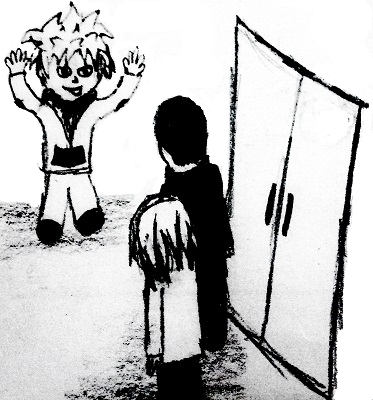
standard input

output

standard output

*Note that girls in Arpa’s land are really attractive.*

Arpa loves overnight parties. In the middle of one of these parties Mehrdad suddenly appeared. He saw *n* pairs of friends sitting around a table. *i*-th pair consisted of a boy, sitting on the *ai*-th chair, and his girlfriend, sitting on the *bi*-th chair. The chairs were numbered 1through 2*n* in clockwise direction. There was exactly one person sitting on each chair.



There were two types of food: Kooft and Zahre-mar. Now Mehrdad wonders, was there any way to serve food for the guests such that:

* Each person had exactly one type of food,
* No boy had the same type of food as his girlfriend,
* Among any three guests sitting on consecutive chairs, there was two of them who had different type of food. Note that chairs 2*n* and 1 are considered consecutive.

Find the answer for the Mehrdad question. If it was possible, find some arrangement of food types that satisfies the conditions.

**Input**

The first line contains an integer *n* (1  ≤  *n*  ≤  105) — the number of pairs of guests.

The *i*-th of the next *n* lines contains a pair of integers *ai* and *bi* (1  ≤ *ai*, *bi* ≤  2*n*) — the number of chair on which the boy in the *i*-th pair was sitting and the number of chair on which his girlfriend was sitting. It's guaranteed that there was exactly one person sitting on each chair.

**Output**

If there is no solution, print -1.

Otherwise print *n* lines, the *i*-th of them should contain two integers which represent the type of food for the *i*-th pair. The first integer in the line is the type of food the boy had, and the second integer is the type of food the girl had. If someone had Kooft, print 1, otherwise print 2.

If there are multiple solutions, print any of them.

**Example**

**input**

3  
1 4  
2 5  
3 6

**output**

1 2  
2 1  
1 2

#include<iostream>

#include<string>

#include<vector>

#include<algorithm>

#include<queue>

#include<cmath>

#include<stack>

#include<climits>

#include<map>

#include<set>

#include<utility>

#include<iterator>

#include<cstring>

#include<cstdio>

#include<unordered\_map>

#define MOD 1000000007

#define pb push\_back

#define mp make\_pair

#define umap unordered\_map

#define ff first

#define ss second

//#define scd static\_cast<double>

using namespace std;

using ll=long long;

using ull=unsigned long long;

using pii=pair<int, int>;

using pll=pair<ll,ll>;

using vi=vector<int>;

using vll=vector<ll>;

using pill=pair<int,ll>;

using vvi=vector<vi>;

ll gcd(ll a, ll b)

{

ll t;

while((t=a%b)!=0) {

a=b;

b=t;

}

return b;

}

ll fast\_exp(ll base, ll n, ll m)

{

if(n==0) return 1;

ll t=fast\_exp(base,n/2,m);

if(n%2==0) return (t\*t)%m;

else return (((t\*t)%m)\*base)%m;

}

bool is\_set(int i, ll mask)

{

return (mask>>i)&1;

}

int count\_bits(ll mask)

{

int ans=0;

for(int i=0;i<64;++i)

if (is\_set(i,mask)) ++ans;

return ans;

}

int first\_bit(ll mask)

{

int i=0;

while(i<64)

if(is\_set(i++, mask)) return i-1;

return -1;

}

vvi g(200001);

int col[200001];

pii p[100001];

void dfs(int x, int c)

{

col[x] = c;

for (auto i:g[x]) if (!col[i]) dfs(i,1+(c==1));

}

int main()

{

ios::sync\_with\_stdio(false);

int n;cin>>n;

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

int a, b; cin>>a>>b;

p[i] = {a,b};

g[a].pb(b); g[b].pb(a);

g[2\*i].pb(2\*i-1); g[2\*i-1].pb(2\*i);

}

for (int i=1; i<=2\*n; ++i) if (!col[i]) dfs(i,1);

for (int i=1; i<=n; ++i) cout << col[p[i].ff] << " " << col[p[i].ss] << "\n";

return 0;

}

<http://codeforces.com/contest/1/problem/B>

B. Spreadsheets

time limit per test

10 seconds

memory limit per test

64 megabytes

input

standard input

output

standard output

In the popular spreadsheets systems (for example, in Excel) the following numeration of columns is used. The first column has number A, the second — number B, etc. till column 26 that is marked by Z. Then there are two-letter numbers: column 27 has number AA, 28 — AB, column 52 is marked by AZ. After ZZ there follow three-letter numbers, etc.

The rows are marked by integer numbers starting with 1. The cell name is the concatenation of the column and the row numbers. For example, BC23 is the name for the cell that is in column 55, row 23.

Sometimes another numeration system is used: RXCY, where X and Y are integer numbers, showing the column and the row numbers respectfully. For instance, R23C55 is the cell from the previous example.

Your task is to write a program that reads the given sequence of cell coordinates and produce each item written according to the rules of another numeration system.

**Input**

The first line of the input contains integer number *n* (1 ≤ *n* ≤ 105), the number of coordinates in the test. Then there follow *n* lines, each of them contains coordinates. All the coordinates are correct, there are no cells with the column and/or the row numbers larger than 106.

**Output**

Write *n* lines, each line should contain a cell coordinates in the other numeration system.

**Examples**

**input**

2  
R23C55  
BC23

**output**

BC23  
R23C55

#include <stdio.h>

#include <stdlib.h>

#define DEBUG 1

int isNum(char a){

return '0'<=a&&a<='9';

}

void reverseString(char a[],int n){

for(int i=0;i<n/2;i++){

int temp = a[i];

a[i]=a[n-1-i];

a[n-1-i]=temp;

}

}

int main(){

int total;

scanf("%d",&total);

char input[100];

for(int i = 0; i < total; i++){

scanf("%s",input);

int type=-1;

int index=1;

int row = 0,column = 0;

int length;

while(input[index]!='\0'){

index++;

}

length = index ;

index= 1;

while(input[index]!='\0'){

if(isNum(input[index])&&!isNum(input[index-1])){

type\*=-1;

}

index+=1;

}

// #if DEBUG

// printf("%d\n", type);

// #endif

if(type==-1){

int start1,end1,start2,end2;

int f1=0;

index = 0;

while(input[index]!='\0'){

if(isNum(input[index])&&!isNum(input[index-1]) &&!f1){

start1 = index;

f1=1;

}

if(!isNum(input[index])&&isNum(input[index-1])){

end1 = index-1;

break;

}

index++;

}

for(int i = start1; i<=end1; i++){

row = 10\*row+(input[i]-'0');

}

start2 = end1+2;

end2 = length-1;

for(int i = start2; i<=end2; i++){

column = 10\*column+(input[i]-'0');

}

char c[100];int l=0;

while(column!=0){

if(column%26 == 0){

c[l] = 'Z';

column/=26;

column-=1;

}

else{

c[l]=column%26+'A'-1;

column/=26;

}

l++;

}

c[l]='\0';

reverseString(c,l);

printf("%s%d\n",c,row);

}else{

int start1=0,end1,start2,end2=length-1;

index = 1;

while(input[index]!='\0'){

if(isNum(input[index])&&!isNum(input[index-1]) ){

start2 = index;

end1 = start2 - 1;

}

index+=1;

}

for(int i = start2; i<=end2; i++){

row = 10\*row+(input[i]-'0');

}

index = start1;

while(index != start2){

column = column\*26 + (input[index] - 'A' +1);

index+=1;

}

printf("R%dC%d\n",row,column );

}

}

}

<http://codeforces.com/contest/469/problem/A>

A. I Wanna Be the Guy

time limit per test

1 second

memory limit per test

256 megabytes

input

standard input

output

standard output

There is a game called "I Wanna Be the Guy", consisting of *n* levels. Little X and his friend Little Y are addicted to the game. Each of them wants to pass the whole game.

Little X can pass only *p* levels of the game. And Little Y can pass only *q* levels of the game. You are given the indices of levels Little X can pass and the indices of levels Little Y can pass. Will Little X and Little Y pass the whole game, if they cooperate each other?

**Input**

The first line contains a single integer *n* (1 ≤  *n* ≤ 100).

The next line contains an integer *p* (0 ≤ *p* ≤ *n*) at first, then follows *p* distinct integers *a*1, *a*2, ..., *ap* (1 ≤ *ai* ≤ *n*). These integers denote the indices of levels Little X can pass. The next line contains the levels Little Y can pass in the same format. It's assumed that levels are numbered from 1 to *n*.

**Output**

If they can pass all the levels, print "I become the guy.". If it's impossible, print "Oh, my keyboard!" (without the quotes).

**Examples**

**input**

4  
3 1 2 3  
2 2 4

**output**

I become the guy.

**input**

4  
3 1 2 3  
2 2 3

**output**

Oh, my keyboard!

**Note**

In the first sample, Little X can pass levels [1 2 3], and Little Y can pass level [2 4], so they can pass all the levels both.

In the second sample, no one can pass level 4.

#include <bits/stdc++.h>

using namespace std;

int n,a,b;

int chk[100];

int main()

{

cin>>n;

cin>>a;

int ch[a];

for(int i=0; i<a; i++)

{

cin>>ch[i];

chk[ch[i]-1]=1;

}

cin>>b;

int c[b];

for(int i=0; i<b; i++)

{

cin>>c[i];

chk[c[i]-1]=1;

}

for(int i=0; i<n; i++)

{

if(chk[i]==0)

{

cout<<"Oh, my keyboard!";

return 0;

}

}

cout<<"I become the guy.";

return 0;

}

<http://codeforces.com/contest/489/problem/C>

C. Given Length and Sum of Digits...

time limit per test

1 second

memory limit per test

256 megabytes

input

standard input

output

standard output

You have a positive integer *m* and a non-negative integer *s*. Your task is to find the smallest and the largest of the numbers that have length *m* and sum of digits *s*. The required numbers should be non-negative integers written in the decimal base without leading zeroes.

**Input**

The single line of the input contains a pair of integers *m*, *s* (1 ≤ *m* ≤ 100, 0 ≤ *s* ≤ 900) — the length and the sum of the digits of the required numbers.

**Output**

In the output print the pair of the required non-negative integer numbers — first the minimum possible number, then — the maximum possible number. If no numbers satisfying conditions required exist, print the pair of numbers "-1 -1" (without the quotes).

**Examples**

**input**

2 15

**output**

69 96

**input**

3 0

**output**

-1 -1

#include <iostream>

#include<vector>

using namespace std;

int main()

{

int i,j,f=0,m,s;

vector<int> l,v;

cin>>m>>s;

j=m;

if(9\*m<s)

{

cout<<-1<<" "<<-1;

return 0;

}

if(m>1&&s<=0)

{cout<<-1<<" "<<-1;

return 0;

}

int t=m,k=s;

while(m--)

{

if(s/10>0)

{

v.push\_back(9);

s-=9;

}

else

{

v.push\_back(s%10);

s=0;

}

}

while(j--)

{

if(k/10>0)

{

l.push\_back(9);

k-=9;

}

else

{

l.push\_back(k%10);

k=0;

}

if(k==0)

{

if(j!=0)

f=1;

}

}

for(i=l.size()-1;i>=0;i--)

{

if(i==l.size()-1)

{

if(f)

cout<<l[i]+1;

else

cout<<l[i];

}

else

{

if(l[i]==0)

cout<<l[i];

else

{

if(f)

{ cout<<l[i]-1;

f=0;

}

else

cout<<l[i];

}

}

}

cout<<" ";

for(i=0;i<v.size();i++)

cout<<v[i];

// your code goes here

return 0;

}

<http://codeforces.com/contest/231/problem/A>

A. Team

time limit per test

2 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

One day three best friends Petya, Vasya and Tonya decided to form a team and take part in programming contests. Participants are usually offered several problems during programming contests. Long before the start the friends decided that they will implement a problem if at least two of them are sure about the solution. Otherwise, the friends won't write the problem's solution.

This contest offers *n* problems to the participants. For each problem we know, which friend is sure about the solution. Help the friends find the number of problems for which they will write a solution.

**Input**

The first input line contains a single integer *n* (1 ≤ *n* ≤ 1000) — the number of problems in the contest. Then *n* lines contain three integers each, each integer is either 0 or 1. If the first number in the line equals 1, then Petya is sure about the problem's solution, otherwise he isn't sure. The second number shows Vasya's view on the solution, the third number shows Tonya's view. The numbers on the lines are separated by spaces.

**Output**

Print a single integer — the number of problems the friends will implement on the contest.

**Examples**

**input**

3  
1 1 0  
1 1 1  
1 0 0

**output**

2

**input**

2  
1 0 0  
0 1 1

**output**

1

**Note**

In the first sample Petya and Vasya are sure that they know how to solve the first problem and all three of them know how to solve the second problem. That means that they will write solutions for these problems. Only Petya is sure about the solution for the third problem, but that isn't enough, so the friends won't take it.

In the second sample the friends will only implement the second problem, as Vasya and Tonya are sure about the solution.

#include<iostream>

using namespace std;

int main(){

int n;

while(cin>>n){

int x,y,z;

int ct = 0;

for(int i=0;i<n;i++){

cin>>x>>y>>z;

if((x+y+z)>1){

ct++;

}

}

cout<<ct<<endl;

}

}

<http://codeforces.com/contest/467/problem/A>

A. George and Accommodation

time limit per test

1 second

memory limit per test

256 megabytes

input

standard input

output

standard output

George has recently entered the BSUCP (Berland State University for Cool Programmers). George has a friend Alex who has also entered the university. Now they are moving into a dormitory.

George and Alex want to live in the same room. The dormitory has *n* rooms in total. At the moment the *i*-th room has *pi* people living in it and the room can accommodate *qi* people in total (*pi* ≤ *qi*). Your task is to count how many rooms has free place for both George and Alex.

**Input**

The first line contains a single integer *n* (1 ≤ *n* ≤ 100) — the number of rooms.

The *i*-th of the next *n* lines contains two integers *pi* and *qi* (0 ≤ *pi* ≤ *qi* ≤ 100) — the number of people who already live in the *i*-th room and the room's capacity.

**Output**

Print a single integer — the number of rooms where George and Alex can move in.

**Examples**

**input**

3  
1 1  
2 2  
3 3

**output**

0

**input**

3  
1 10  
0 10  
10 10

**output**

2

#include<bits/stdc++.h>

using namespace std;

int main()

{

int cc=0;

int n;

cin>>n;

for(int i=0; i<n; i++)

{

vector<int> v;

for(int i=0; i<2; i++)

{

int t;

cin>>t;

v. push\_back(t);

}

if((v[1]-v[0])>=2)cc++;

}

cout<<cc<<endl;

return 0;

}

<http://codeforces.com/contest/409/problem/H>

H. A + B Strikes Back

time limit per test

1 second

memory limit per test

256 megabytes

input

standard input

output

standard output

A + B is often used as an example of the easiest problem possible to show some contest platform. However, some scientists have observed that sometimes this problem is not so easy to get accepted. Want to try?

**Input**

The input contains two integers *a* and *b* (0 ≤ *a*, *b* ≤ 103), separated by a single space.

**Output**

Output the sum of the given integers.

**Examples**

**input**

5 14

**output**

19

**input**

381 492

**output**

873

#include <bits/stdc++.h>

using namespace std;

#define FOR(i,a,b) for(long long int i = a; i < b; i++)

#define mp make\_pair

#define pb push\_back

#define st first

#define nd second

typedef pair <int , int> pii;

typedef long long ll;

int main(){

int a,b;

cin >> a >> b;

cout << a+b << endl;

//bamaze :/

}

<http://codeforces.com/contest/136/problem/A>

A. Presents

time limit per test

2 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

Little Petya very much likes gifts. Recently he has received a new laptop as a New Year gift from his mother. He immediately decided to give it to somebody else as what can be more pleasant than giving somebody gifts. And on this occasion he organized a New Year party at his place and invited *n* his friends there.

If there's one thing Petya likes more that receiving gifts, that's watching others giving gifts to somebody else. Thus, he safely hid the laptop until the next New Year and made up his mind to watch his friends exchanging gifts while he does not participate in the process. He numbered all his friends with integers from 1 to *n*. Petya remembered that a friend number *i* gave a gift to a friend number *pi*. He also remembered that each of his friends received exactly one gift.

Now Petya wants to know for each friend *i* the number of a friend who has given him a gift.

**Input**

The first line contains one integer *n* (1 ≤ *n* ≤ 100) — the quantity of friends Petya invited to the party. The second line contains *n* space-separated integers: the *i*-th number is *pi* — the number of a friend who gave a gift to friend number *i*. It is guaranteed that each friend received exactly one gift. It is possible that some friends do not share Petya's ideas of giving gifts to somebody else. Those friends gave the gifts to themselves.

**Output**

Print *n* space-separated integers: the *i*-th number should equal the number of the friend who gave a gift to friend number *i*.

**Examples**

**input**

4  
2 3 4 1

**output**

4 1 2 3

**input**

3  
1 3 2

**output**

1 3 2

**input**

2  
1 2

**output**

1 2

#include <stdio.h>

int main ()

{

int n,i,t;

int f[100];

scanf("%d",&n);

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

{

scanf("%d",&t);

f[t]=i;

}

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

printf("%d ",f[i]);

return 0;

}

<http://codeforces.com/contest/6/problem/C>

C. Alice, Bob and Chocolate

time limit per test

2 seconds

memory limit per test

64 megabytes

input

standard input

output

standard output

Alice and Bob like games. And now they are ready to start a new game. They have placed *n* chocolate bars in a line. Alice starts to eat chocolate bars one by one from left to right, and Bob — from right to left. For each chocololate bar the time, needed for the player to consume it, is known (Alice and Bob eat them with equal speed). When the player consumes a chocolate bar, he immediately starts with another. It is not allowed to eat two chocolate bars at the same time, to leave the bar unfinished and to make pauses. If both players start to eat the same bar simultaneously, Bob leaves it to Alice as a true gentleman.

How many bars each of the players will consume?

**Input**

The first line contains one integer *n* (1 ≤ *n* ≤ 105) — the amount of bars on the table. The second line contains a sequence *t*1, *t*2, ..., *tn*(1 ≤ *ti* ≤ 1000), where *ti* is the time (in seconds) needed to consume the *i*-th bar (in the order from left to right).

**Output**

Print two numbers *a* and *b*, where *a* is the amount of bars consumed by Alice, and *b* is the amount of bars consumed by Bob.

**Examples**

**input**

5  
2 9 8 2 7

**output**

2 3

]#include <bits/stdc++.h>

using namespace std;

vector <int> v;

int ar[100010] , dp[100010] , dp2[100010];

int main()

{

int n;

cin >> n;

for(int i = 0; i < n; i++)

{

cin >> ar[i];

dp[i+1] = ar[i] + dp[i];

}

for(int i = n-2; i >= 0; i--)

dp2[i] = ar[i+1] + dp2[i+1];

for(int i = 0; i < n; i++)

{

if(dp[i] > dp2[i])

{

cout << i << " " << n-i;

return 0;

}

}

cout << 1 << " " << 0;

return 0;

}

<http://codeforces.com/contest/743/problem/A>

A. Vladik and flights

time limit per test

2 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

Vladik is a competitive programmer. This year he is going to win the International Olympiad in Informatics. But it is not as easy as it sounds: the question Vladik face now is to find the cheapest way to get to the olympiad.

Vladik knows *n* airports. All the airports are located on a straight line. Each airport has unique id from 1 to *n*, Vladik's house is situated next to the airport with id *a*, and the place of the olympiad is situated next to the airport with id *b*. It is possible that Vladik's house and the place of the olympiad are located near the same airport.

To get to the olympiad, Vladik can fly between any pair of airports any number of times, but he has to start his route at the airport *a* and finish it at the airport *b*.

Each airport belongs to one of two companies. The cost of flight from the airport *i* to the airport *j* is zero if both airports belong to the same company, and |*i* - *j*| if they belong to different companies.

Print the minimum cost Vladik has to pay to get to the olympiad.

**Input**

The first line contains three integers *n*, *a*, and *b* (1 ≤ *n* ≤ 105, 1 ≤ *a*, *b* ≤ *n*) — the number of airports, the id of the airport from which Vladik starts his route and the id of the airport which he has to reach.

The second line contains a string with length *n*, which consists only of characters 0 and 1. If the *i*-th character in this string is 0, then *i*-th airport belongs to first company, otherwise it belongs to the second.

**Output**

Print single integer — the minimum cost Vladik has to pay to get to the olympiad.

**Examples**

**input**

4 1 4  
1010

**output**

1

**input**

5 5 2  
10110

**output**

0

**Note**

In the first example Vladik can fly to the airport 2 at first and pay |1 - 2| = 1 (because the airports belong to different companies), and then fly from the airport 2 to the airport 4 for free (because the airports belong to the same company). So the cost of the whole flight is equal to 1. It's impossible to get to the olympiad for free, so the answer is equal to 1.

In the second example Vladik can fly directly from the airport 5 to the airport 2, because they belong to the same company.

#include <bits/stdc++.h>

using namespace std;

#define fi first

#define se second

#define mp make\_pair

#define pb push\_back

#define pf push\_front;

#define fbo find\_by\_order

#define ook order\_of\_key

#define lb lower\_bound

#define ub upper\_bound

#define rep(i,n) for(int i=0;i<n;i++)

#define fori(a,b) for(int i=a;i<=b;i++)

#define mem(x,a) memset(x,a,sizeof(x))

typedef long long ll;

typedef pair<int,int> ii;

typedef vector<ll> vi;

typedef long double ld;

typedef map<ll,ll> spt;

typedef set<ll> si;

typedef multiset<ll>::iterator sit;

typedef map<int,int>::iterator mi;

typedef vector<int>::iterator vit;

typedef vector<ii> vii;

typedef set<ii> sii;

typedef multiset<ll> msi;

typedef vector< vector<ll> > matrix;

const ll INF = 1e18;

const int MOD = 1e9 + 7;

const int N = 1e6 ;

ll modpow(ll a, ll b, ll c) {

int ans=1;

while(b != 0) {

if(b%2 == 1)

ans=(ans\*a)%c;

a=(a\*a)%c;

b /= 2;

}

return ans;

}

bool cmp(ii x,ii y)

{

if(abs(x.fi)==abs(y.fi))

return abs(x.se)<=abs(y.se);

return abs(x.fi)<abs(y.fi);

}

int main(){

//freopen("double\_squares.txt","r",stdin);

//freopen("aoutout.txt","w",stdout);

ios\_base::sync\_with\_stdio(0);

cin.tie(0);

cout.tie(0);

//ll n,a,b,q,e,f,k,flag=-1,ans=0;

int n, a, b;

string s;

cin >> n >> a >> b >> s;

a--, b--;

cout << ((s[a] - '0') ^ (s[b] - '0'));

}

<http://codeforces.com/contest/519/problem/B>

B. A and B and Compilation Errors

time limit per test

2 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

*A and B are preparing themselves for programming contests.*

B loves to debug his code. But before he runs the solution and starts debugging, he has to first compile the code.

Initially, the compiler displayed *n* compilation errors, each of them is represented as a positive integer. After some effort, B managed to fix some mistake and then another one mistake.

However, despite the fact that B is sure that he corrected the two errors, he can not understand exactly what compilation errors disappeared — the compiler of the language which B uses shows errors in the new order every time! B is sure that unlike many other programming languages, compilation errors for his programming language do not depend on each other, that is, if you correct one error, the set of other error does not change.

Can you help B find out exactly what two errors he corrected?

**Input**

The first line of the input contains integer *n* (3 ≤ *n* ≤ 105) — the initial number of compilation errors.

The second line contains *n* space-separated integers *a*1, *a*2, ..., *an* (1 ≤ *ai* ≤ 109) — the errors the compiler displayed for the first time.

The third line contains *n* - 1 space-separated integers *b*1, *b*2, ..., *bn*- 1 — the errors displayed at the second compilation. It is guaranteed that the sequence in the third line contains all numbers of the second string except for exactly one.

The fourth line contains *n* - 2 space-separated integers *с*1, *с*2, ..., *сn*- 2 — the errors displayed at the third compilation. It is guaranteed that the sequence in the fourth line contains all numbers of the third line except for exactly one.

**Output**

Print two numbers on a single line: the numbers of the compilation errors that disappeared after B made the first and the second correction, respectively.

**Examples**

**input**

5  
1 5 8 123 7  
123 7 5 1  
5 1 7

**output**

8  
123

**input**

6  
1 4 3 3 5 7  
3 7 5 4 3  
4 3 7 5

**output**

1  
3

**Note**

In the first test sample B first corrects the error number 8, then the error number 123.

In the second test sample B first corrects the error number 1, then the error number 3. Note that if there are multiple errors with the same number, B can correct only one of them in one step.

#include <bits/stdc++.h>

#define ll long long

using namespace std;

int main(){

ll n, a, sum = 0, sum1 = 0, sum2 = 0;

cin >> n;

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

cin >> a;

sum += a;

}

for(int i = 1; i <= n - 1; i++){

cin >> a;

sum1 += a;

}

for(int i = 1; i <= n - 2; i++){

cin >> a;

sum2 += a;

}

cout << sum - sum1 <<endl;

cout << sum1 - sum2;

}

<http://codeforces.com/contest/230/problem/B>

B. T-primes

time limit per test

2 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

We know that prime numbers are positive integers that have exactly two distinct positive divisors. Similarly, we'll call a positive integer *tТ-prime*, if *t* has exactly three distinct positive divisors.

You are given an array of *n* positive integers. For each of them determine whether it is Т-prime or not.

**Input**

The first line contains a single positive integer, *n* (1 ≤ *n* ≤ 105), showing how many numbers are in the array. The next line contains *n*space-separated integers *xi* (1 ≤ *xi* ≤ 1012).

Please, do not use the %lld specifier to read or write 64-bit integers in С++. It is advised to use the cin, cout streams or the %I64dspecifier.

**Output**

Print *n* lines: the *i*-th line should contain "YES" (without the quotes), if number *xi* is Т-prime, and "NO" (without the quotes), if it isn't.

**Examples**

**input**

3  
4 5 6

**output**

YES  
NO  
NO

**Note**

The given test has three numbers. The first number 4 has exactly three divisors — 1, 2 and 4, thus the answer for this number is "YES". The second number 5 has two divisors (1 and 5), and the third number 6 has four divisors (1, 2, 3, 6), hence the answer for them is "NO".

#include <bits/stdc++.h>

using namespace std;

bool prime[1000001];

bool IsPerfectSquare(long long input)

{

return sqrt(input)==(long long) sqrt(input);

}

void Intilize()

{

memset(prime,true,(int)pow(10,6)+1);

for(int i=2; i\*i<=(int)pow(10,6); i++)

{

if(prime[i]==true)

{

for(int j=2\*i; j<=(int)pow(10,6); j+=i)

{

prime[j]=false;

}

}

}

}

void Prime(long long number)

{

if(number!=1&&IsPerfectSquare(number)&&prime[(long)sqrt(number)])

{

cout<<"YES"<<endl;

return;

}

cout << "NO" << endl;

return;

}

int main()

{

Intilize();

int number;

cin >> number;

long long arr[number];

for(int i=0; i<number; i++)

{

cin >> arr[i];

Prime(arr[i]);

}

return 0;

}

<http://codeforces.com/contest/2/problem/A>

A. Winner

time limit per test

1 second

memory limit per test

64 megabytes

input

standard input

output

standard output

The winner of the card game popular in Berland "Berlogging" is determined according to the following rules. If at the end of the game there is only one player with the maximum number of points, he is the winner. The situation becomes more difficult if the number of such players is more than one. During each round a player gains or loses a particular number of points. In the course of the game the number of points is registered in the line "name score", where name is a player's name, and score is the number of points gained in this round, which is an integer number. If score is negative, this means that the player has lost in the round. So, if two or more players have the maximum number of points (say, it equals to *m*) at the end of the game, than wins the one *of them* who scored at least *m* points first. Initially each player has 0 points. It's guaranteed that at the end of the game at least one player has a positive number of points.

**Input**

The first line contains an integer number *n* (1  ≤  *n*  ≤  1000), *n* is the number of rounds played. Then follow *n* lines, containing the information about the rounds in "name score" format in chronological order, where name is a string of lower-case Latin letters with the length from 1 to 32, and score is an integer number between -1000 and 1000, inclusive.

**Output**

Print the name of the winner.

**Examples**

**input**

3  
mike 3  
andrew 5  
mike 2

**output**

andrew

**input**

3  
andrew 3  
andrew 2  
mike 5

**output**

andrew

#include <iostream>

#include <map>

#include <string>

using namespace std;

int n, m, arrid[1005], arrpoin[1005], high, poin[1005], poin2[1005];

string nama[1005];

map<string,int> pt;

int main(){

cin>>n;

for(int i = 0; i < n; i++ ){

string name; cin>>name>>arrpoin[i];

int idx = -1;

map<string,int>::iterator it = pt.find(name);

if(it!=pt.end()) idx = it->second; else pt[name] = m;

if(idx==-1) idx = m++;

nama[idx] = name, arrid[i] = idx;

poin[idx] += arrpoin[i];

}

high = poin[0];

for(int i = 1; i < m; i++ ) if(poin[i]>high) high = poin[i];

int winner;

for(int i = 0; i < n; i++ ){

int cur = arrid[i];

if(high==poin[cur]){

poin2[cur] += arrpoin[i];

if(poin2[cur]>=high){ winner = cur; break; }

}

}

cout<<nama[winner]<<'\n';

return 0;

}

<http://codeforces.com/contest/507/problem/A>

A. Amr and Music

time limit per test

1 second

memory limit per test

256 megabytes

input

standard input

output

standard output

Amr is a young coder who likes music a lot. He always wanted to learn how to play music but he was busy coding so he got an idea.

Amr has *n* instruments, it takes *ai* days to learn *i*-th instrument. Being busy, Amr dedicated *k* days to learn how to play the maximum possible number of instruments.

Amr asked for your help to distribute his free days between instruments so that he can achieve his goal.

**Input**

The first line contains two numbers *n*, *k* (1 ≤ *n* ≤ 100, 0 ≤ *k* ≤ 10 000), the number of instruments and number of days respectively.

The second line contains *n* integers *ai* (1 ≤ *ai* ≤ 100), representing number of days required to learn the *i*-th instrument.

**Output**

In the first line output one integer *m* representing the maximum number of instruments Amr can learn.

In the second line output *m* space-separated integers: the indices of instruments to be learnt. You may output indices in any order.

if there are multiple optimal solutions output any. It is not necessary to use all days for studying.

**Examples**

**input**

4 10  
4 3 1 2

**output**

4  
1 2 3 4

**input**

5 6  
4 3 1 1 2

**output**

3  
1 3 4

**input**

1 3  
4

**output**

0

**Note**

In the first test Amr can learn all 4 instruments.

In the second test other possible solutions are: {2, 3, 5} or {3, 4, 5}.

In the third test Amr doesn't have enough time to learn the only presented instrument.

#include<iostream>

using namespace std;

int main()

{

int n,k;

cin>>n>>k;

int a[n],b[n],s=0,no=0;

for(int i=0;i<n;i++)

{

cin>>a[i];

b[i]=i+1;

}

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

{

for(int j=0;j<n-1;j++)

{

if(a[j]>a[j+1]){

int t=a[j];

a[j]=a[j+1];

a[j+1]=t;

t=b[j];

b[j]=b[j+1];

b[j+1]=t;

}

}

}

for(int i=0;i<n;i++)

{

if(s+a[i]<=k){

s=s+a[i];

no++;

}else

break;

}

cout<<no;

if(no>0)

cout<<endl;

for(int i=0;i<no;i++)

{

cout<<b[i];

if(i!=no-1)

cout<<" ";

}

}

<http://codeforces.com/contest/281/problem/A>

A. Word Capitalization

time limit per test

2 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

Capitalization is writing a word with its first letter as a capital letter. Your task is to capitalize the given word.

Note, that during capitalization all the letters except the first one remains unchanged.

**Input**

A single line contains a non-empty word. This word consists of lowercase and uppercase English letters. The length of the word will not exceed 103.

**Output**

Output the given word after capitalization.

**Examples**

**input**

ApPLe

**output**

ApPLe

**input**

konjac

**output**

Konjac

#include<iostream>

#include<bits/stdc++.h>

using namespace std;

int main()

{

string s;

cin>>s;

s[0]=toupper(s[0]);

cout<<s<<endl;

return 0;

}

<http://codeforces.com/contest/617/problem/B>

B. Chocolate

time limit per test

1 second

memory limit per test

256 megabytes

input

standard input

output

standard output

Bob loves everything sweet. His favorite chocolate bar consists of pieces, each piece may contain a nut. Bob wants to break the bar of chocolate into multiple pieces so that each part would contain **exactly** one nut and any break line goes between two adjacent pieces.

You are asked to calculate the number of ways he can do it. Two ways to break chocolate are considered distinct if one of them contains a break between some two adjacent pieces and the other one doesn't.

Please note, that if Bob doesn't make any breaks, all the bar will form one piece and it still has to have exactly one nut.

**Input**

The first line of the input contains integer *n* (1 ≤ *n* ≤ 100) — the number of pieces in the chocolate bar.

The second line contains *n* integers *ai* (0 ≤ *ai* ≤ 1), where 0 represents a piece without the nut and 1 stands for a piece with the nut.

**Output**

Print the number of ways to break the chocolate into multiple parts so that each part would contain exactly one nut.

**Examples**

**input**

3  
0 1 0

**output**

1

**input**

5  
1 0 1 0 1

**output**

4

**Note**

In the first sample there is exactly one nut, so the number of ways equals 1 — Bob shouldn't make any breaks.

In the second sample you can break the bar in four ways:

10|10|1

1|010|1

10|1|01

1|01|01

#include <bits/stdc++.h>

#define endl '\n'

using namespace std;

typedef long long int ll;

ll n , result = 0;

vector<ll> res , ans;

const int MAXN = 1e2+20;

ll a[MAXN];

int main()

{

ios\_base::sync\_with\_stdio(false); cin.tie(0); cout.tie(0);

//Tag Combinatorics Baraye Asle Zarb????

cin >> n;

for(ll i = 0; i < n; i++)

cin >> a[i];

for(ll i = 0; i < n; i++)

if(a[i] == 1)

res.push\_back(i);

ll c = res.size();

if(c > 0){

result = 1;

for(ll i = 0; i < c-1; i++)

ans.push\_back(res[i+1]-res[i]-1);

for(ll i = 0; i < ans.size(); i++)

result \*= ans[i]+1;

}

cout << result << endl;

return 0;

}

<http://codeforces.com/contest/58/problem/A>

A. Chat room

time limit per test

1 second

memory limit per test

256 megabytes

input

standard input

output

standard output

Vasya has recently learned to type and log on to the Internet. He immediately entered a chat room and decided to say hello to everybody. Vasya typed the word *s*. It is considered that Vasya managed to say hello if several letters can be deleted from the typed word so that it resulted in the word "hello". For example, if Vasya types the word "ahhellllloou", it will be considered that he said hello, and if he types "hlelo", it will be considered that Vasya got misunderstood and he didn't manage to say hello. Determine whether Vasya managed to say hello by the given word *s*.

**Input**

The first and only line contains the word *s*, which Vasya typed. This word consisits of small Latin letters, its length is no less that 1 and no more than 100 letters.

**Output**

If Vasya managed to say hello, print "YES", otherwise print "NO".

**Examples**

**input**

ahhellllloou

**output**

YES

**input**

hlelo

**output**

NO

#include<stdio.h>

#include<string.h>

int main()

{

char s[101];

int c=0,i;

scanf("%s",s);

for (i=0;i<strlen(s);i++){

if (c==0 && s[i]=='h')

c++;

else if (c==1&&s[i]=='e')

c++;

else if ((c==2||c==3)&&s[i]=='l')

c++;

else if (c==4&&s[i]=='o')

c++;

}

if (c==5)

printf("YES\n");

else

printf("NO\n");

return 0;

}

<http://codeforces.com/contest/698/problem/A>

A. Vacations

time limit per test

1 second

memory limit per test

256 megabytes

input

standard input

output

standard output

Vasya has *n* days of vacations! So he decided to improve his IT skills and do sport. Vasya knows the following information about each of this *n* days: whether that gym opened and whether a contest was carried out in the Internet on that day. For the *i*-th day there are four options:

1. on this day the gym is closed and the contest is not carried out;
2. on this day the gym is closed and the contest is carried out;
3. on this day the gym is open and the contest is not carried out;
4. on this day the gym is open and the contest is carried out.

On each of days Vasya can either have a rest or write the contest (if it is carried out on this day), or do sport (if the gym is open on this day).

Find the minimum number of days on which Vasya will have a rest (it means, he will not do sport and write the contest at the same time). The only limitation that Vasya has — *he does not want to do the same activity on two consecutive days: it means, he will not do sport on two consecutive days, and write the contest on two consecutive days*.

**Input**

The first line contains a positive integer *n* (1 ≤ *n* ≤ 100) — the number of days of Vasya's vacations.

The second line contains the sequence of integers *a*1, *a*2, ..., *an* (0 ≤ *ai* ≤ 3) separated by space, where:

* *ai* equals 0, if on the *i*-th day of vacations the gym is closed and the contest is not carried out;
* *ai* equals 1, if on the *i*-th day of vacations the gym is closed, but the contest is carried out;
* *ai* equals 2, if on the *i*-th day of vacations the gym is open and the contest is not carried out;
* *ai* equals 3, if on the *i*-th day of vacations the gym is open and the contest is carried out.

**Output**

Print the minimum possible number of days on which Vasya will have a rest. Remember that Vasya refuses:

* to do sport on any two consecutive days,
* to write the contest on any two consecutive days.

**Examples**

**input**

4  
1 3 2 0

**output**

2

**input**

7  
1 3 3 2 1 2 3

**output**

0

**input**

2  
2 2

**output**

1

**Note**

In the first test Vasya can write the contest on the day number 1 and do sport on the day number 3. Thus, he will have a rest for only 2 days.

In the second test Vasya should write contests on days number 1, 3, 5 and 7, in other days do sport. Thus, he will not have a rest for a single day.

In the third test Vasya can do sport either on a day number 1 or number 2. He can not do sport in two days, because it will be contrary to the his limitation. Thus, he will have a rest for only one day.

#include <cstdlib>

#include <cstdio>

#include <algorithm>

#include <vector>

#include <queue>

#include <cmath>

#include <stack>

#include <map>

#include <set>

#include <deque>

#include <cstring>

#include <functional>

#include <climits>

#include <list>

#include <ctime>

#include <complex>

#define F1(x,y,z) for(int x=(y);x<(z);x++)

#define F2(x,y,z) for(int x=(y);x<=(z);x++)

#define F3(x,y,z) for(int x=(y);x>(z);x--)

#define F4(x,y,z) for(int x=(y);x>=(z);x--)

#define mp make\_pair

#define pb push\_back

#define LL long long

#define co complex<double>

#define fi first

#define se second

#define MAX 100005

#define AMAX 1025\*1005

#define MOD 1000000007

#define f(c,d) ((1<<(c))\*(d))

using namespace std;

int n,ta,la,ans;

int main(){

scanf("%d",&n);

ans=n;

while(n--){

scanf("%d",&ta);

if(ta==0)la=0;

else if(ta==1){

if(la==1)la=0;

else ans--,la=1;

}else if(ta==2){

if(la==2)la=0;

else ans--,la=2;

}else{

ans--;

if(la)la=3-la;

}

}

printf("%d\n",ans);

#ifdef LOCAL\_PROJECT

system("pause");

#endif

return 0;

}

<http://codeforces.com/contest/663/problem/C>

C. Graph Coloring

time limit per test

1 second

memory limit per test

256 megabytes

input

standard input

output

standard output

You are given an undirected graph that consists of *n* vertices and *m* edges. Initially, each edge is colored either red or blue. Each turn a player picks a single vertex and switches the color of **all** edges incident to it. That is, all red edges with an endpoint in this vertex change the color to blue, while all blue edges with an endpoint in this vertex change the color to red.

Find the minimum possible number of moves required to make the colors of all edges equal.

**Input**

The first line of the input contains two integers *n* and *m* (1 ≤ *n*, *m* ≤ 100 000) — the number of vertices and edges, respectively.

The following *m* lines provide the description of the edges, as the *i*-th of them contains two integers *ui* and *vi* (1 ≤ *ui*, *vi* ≤ *n*, *ui* ≠ *vi*) — the indices of the vertices connected by the *i*-th edge, and a character *ci* (http://codeforces.com/predownloaded/dd/1a/dd1a02b63fcdd5724ddc240af201bc41d5f30ce8.png) providing the initial color of this edge. If *ci*equals 'R', then this edge is initially colored red. Otherwise, *ci* is equal to 'B' and this edge is initially colored blue. It's guaranteed that there are no self-loops and multiple edges.

**Output**

If there is no way to make the colors of all edges equal output  - 1 in the only line of the output. Otherwise first output *k* — the minimum number of moves required to achieve the goal, then output *k* integers *a*1, *a*2, ..., *ak*, where *ai* is equal to the index of the vertex that should be used at the *i*-th move.

If there are multiple optimal sequences of moves, output any of them.

**Examples**

**input**

3 3  
1 2 B  
3 1 R  
3 2 B

**output**

1  
2

**input**

6 5  
1 3 R  
2 3 R  
3 4 B  
4 5 R  
4 6 R

**output**

2  
3 4

**input**

4 5  
1 2 R  
1 3 R  
2 3 B  
3 4 B  
1 4 B

**output**

-1

<http://codeforces.com/contest/363/problem/C>

C. Fixing Typos

time limit per test

1 second

memory limit per test

256 megabytes

input

standard input

output

standard output

Many modern text editors automatically check the spelling of the user's text. Some editors even suggest how to correct typos.

In this problem your task to implement a small functionality to correct two types of typos in a word. We will assume that three identical letters together is a typo (for example, word "helllo" contains a typo). Besides, a couple of identical letters immediately followed by another couple of identical letters is a typo too (for example, words "helloo" and "wwaatt" contain typos).

Write a code that deletes the minimum number of letters from a word, correcting described typos in the word. You are allowed to delete letters from both ends and from the middle of the word.

**Input**

The single line of the input contains word *s*, its length is from 1 to 200000 characters. The given word *s* consists of lowercase English letters.

**Output**

Print such word *t* that it doesn't contain any typos described in the problem statement and is obtained from *s* by deleting the least number of letters.

If there are multiple solutions, print any of them.

**Examples**

**input**

helloo

**output**

hello

**input**

woooooow

**output**

woow

**Note**

The second valid answer to the test from the statement is "heloo".

#include<bits/stdc++.h>

#define all(a) (a).begin(), (a).end()

#define allr(a) (a).rbegin(), (a).rend()

#define mmm(k,n) memset(k,n,sizeof k)

#define fast\_io ios\_base::sync\_with\_stdio(false); cin.tie(0)

#define digits(a) fixed << setprecision(a)

#define x first

#define y second

#define oo 1e9

#define pi acos(-1)

#define MOD 1000000007

using namespace std ;

template<typename X> inline X abs(const X& a) { return a < 0? -a: a; }

template<typename X> inline X sqr(const X& a) { return a \* a; }

typedef pair<int, int> ii ;

typedef vector<ii> vii ;

typedef vector<int> vi ;

long long combine(long long n , int k)

{

long long ans =1 ;

for(int i=0 ; i< k ; i++)

ans=(ans \*(n-i))/(i+1) ;

return ans ;

}

long long pw(long long a, long long p)

{

if(p==0)

return 1 ;

if(p==1)

return a ;

long long half=pw(a,p/2) ;

if(p%2==0)

return half\*half ;

return half\*half\*a ;

}

long long dis(pair<long long, long long> a , pair<long long, long long> b)

{

return (a.x-b.x)\*(a.x-b.x) + (a.y-b.y)\*(a.y-b.y) ;

}

long long gcd(long long a, long long b)

{

return b == 0 ? a : gcd(b, a % b);

}

long long lcm(long long a, long long b)

{

return (a\*b)/gcd(a,b) ;

}

bitset<10000001> bs ;

vi sieve()

{

vi primes ;

long long sieve\_size = 10000000;

bs.set();

bs[0] = bs[1] = 0;

for (long long i = 2; i <= sieve\_size; i++) if (bs[i])

{

for (long long j = i \* i; j <=sieve\_size; j += i) bs[j] = 0;

primes.push\_back((int)i);

}

return primes ;

}

vi primes ;

set<int> primeFactors(long long N) { // remember: vi is vector<int>, ll is long long

set<int> factors;

long long PF\_idx = 0, PF = primes[PF\_idx]; // primes has been populated by sieve

while (PF \* PF <= N) { // stop at sqrt(N); N can get smaller

while (N % PF == 0) { N /= PF; factors.insert(PF); } // remove PF

PF = primes[++PF\_idx]; // only consider primes!

}

if (N != 1) factors.insert(N); // special case if N is a prime

return factors; // if N does not fit in 32-bit integer and is a prime

} // then ‘factors’ will have to be changed to vector<ll>

long long gauss(long long n){

return n\*(n+1)/2;

}

int main() {

//freopen("a.in","r",stdin) ;

//freopen("a.out","w", stdout) ;

fast\_io ;

string s, ans="" ;

bool couple =false ;

cin >> s ;

int n =(int)s.size(), l=0 ;

while(l<n){

ans+=s[l] ;

int r = l ;

while(s[l]==s[r] && r<n)

r++ ;

if(r-l==1)

couple = false ;

if(r-l>=2 && !couple){

couple=true ;

ans+=s[l] ;

}

else if(r-l>=2 && couple){

couple=false ;

}

l=r ;

}

cout << ans << endl ;

return 0;

}

<http://codeforces.com/contest/741/problem/C>

C. Arpa’s overnight party and Mehrdad’s silent entering

time limit per test

1 second

memory limit per test

256 megabytes

input

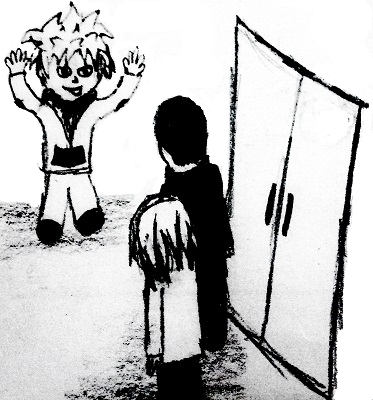
standard input

output

standard output

*Note that girls in Arpa’s land are really attractive.*

Arpa loves overnight parties. In the middle of one of these parties Mehrdad suddenly appeared. He saw *n* pairs of friends sitting around a table. *i*-th pair consisted of a boy, sitting on the *ai*-th chair, and his girlfriend, sitting on the *bi*-th chair. The chairs were numbered 1through 2*n* in clockwise direction. There was exactly one person sitting on each chair.



There were two types of food: Kooft and Zahre-mar. Now Mehrdad wonders, was there any way to serve food for the guests such that:

* Each person had exactly one type of food,
* No boy had the same type of food as his girlfriend,
* Among any three guests sitting on consecutive chairs, there was two of them who had different type of food. Note that chairs 2*n* and 1 are considered consecutive.

Find the answer for the Mehrdad question. If it was possible, find some arrangement of food types that satisfies the conditions.

**Input**

The first line contains an integer *n* (1  ≤  *n*  ≤  105) — the number of pairs of guests.

The *i*-th of the next *n* lines contains a pair of integers *ai* and *bi* (1  ≤ *ai*, *bi* ≤  2*n*) — the number of chair on which the boy in the *i*-th pair was sitting and the number of chair on which his girlfriend was sitting. It's guaranteed that there was exactly one person sitting on each chair.

**Output**

If there is no solution, print -1.

Otherwise print *n* lines, the *i*-th of them should contain two integers which represent the type of food for the *i*-th pair. The first integer in the line is the type of food the boy had, and the second integer is the type of food the girl had. If someone had Kooft, print 1, otherwise print 2.

If there are multiple solutions, print any of them.

**Example**

**input**

3  
1 4  
2 5  
3 6

**output**

1 2  
2 1  
1 2

#include <bits/stdc++.h>

using namespace std;

typedef long long ll;

#define eps 1e-9

#define all(a) a.begin(),a.end()

#define mp make\_pair

#define F first

#define S second

#define pb push\_back

#define sz size()

#define rd(inp) scanf("%lld",&inp)

#define rd2(inp1, inp2) scanf("%lld %lld",&inp1, &inp2)

#define rl(inp) scanf("%d",&inp)

#define pf(out) printf("%lld\n", out);

const long long linf = 1e18+5;

const int mod = (int) 1e9 + 7;

const int inf = 1e9;

ll read(){

bool minus = false;

ll result = 0;

char ch;

ch = getchar();

while (true) {

if (ch == '-') break;

if (ch >= '0' && ch <= '9') break;

ch = getchar();

}

if (ch == '-') minus = true; else result = ch-'0';

while (true) {

ch = getchar();

if (ch < '0' || ch > '9') break;

result = result\*10 + (ch - '0');

}

if (minus){

return -result;

}

else{

return result;

}

}

ll fpow(ll base,ll power){

ll result = 1;

while (power > 0){

if (power%2 == 1) result=(result\*base);

base = (base\*base);

power /= 2;

}

return result;

}

#define maxn 2 \* 100110

ll c[maxn];

vector<ll> adj[maxn];

bool vis[maxn];

void dfs(ll node, ll par){

// cout << node << " dfs\n";

ll j;

vis[node] = true;

if ( c[par] == 1 ){

c[node] = 2;

}

else{

c[node] = 1;

}

for ( j = 0 ; j < adj[node].sz ; j ++ ){

ll to = adj[node][j];

if ( to != par && !vis[to] ){

dfs(to, node);

}

}

}

vector<pair<ll, ll > > edg;

int main(){

ll n, i;

cin >> n;

for ( i = 1 ; i <= 2 \* n ; i += 2 ){

adj[i].pb(i+1);

adj[i+1].pb(i);

}

for ( i = 0 ; i < n ; i ++ ){

ll u, v;

u = read();

v = read();

adj[u].pb(v);

adj[v].pb(u);

edg.pb(mp(u, v));

}

c[0] = 2;

for ( i = 1 ; i <= 2 \* n ; i ++ ){

if ( !vis[i] ){

dfs(i, 0);

}

}

for ( i = 0 ; i < edg.sz ; i ++ ){

printf("%lld %lld\n", c[edg[i].F], c[edg[i].S]);

}

return 0;

}

<http://codeforces.com/contest/452/problem/B>

B. 4-point polyline

time limit per test

2 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

You are given a rectangular grid of lattice points from (0, 0) to (*n*, *m*) inclusive. You have to choose exactly 4 different points to build a polyline possibly with self-intersections and self-touching. This polyline should be as long as possible.

A polyline defined by points *p*1, *p*2, *p*3, *p*4 consists of the line segments *p*1 *p*2, *p*2 *p*3, *p*3 *p*4, and its length is the sum of the lengths of the individual line segments.

**Input**

The only line of the input contains two integers *n* and *m* (0 ≤ *n*, *m* ≤ 1000). It is guaranteed that grid contains at least 4 different points.

**Output**

Print 4 lines with two integers per line separated by space — coordinates of points *p*1, *p*2, *p*3, *p*4 in order which represent the longest possible polyline.

Judge program compares your answer and jury's answer with 10- 6 precision.

**Examples**

**input**

1 1

**output**

1 1  
0 0  
1 0  
0 1

**input**

0 10

**output**

0 1  
0 10  
0 0  
0 9

#include <bits/stdc++.h>

using namespace std;

#define pf push\_front

#define pb push\_back

#define FOR(i,begin,end) for (LL i=begin;i<=end;i++)

#define rep(i,t) for (LL i=0;i<t;i++)

#define pii pair<LL,LL>

#define pi pair<pii,LL>

#define X first

#define Y second

#define fast cin.tie(0),cout.tie(0),ios::sync\_with\_stdio(false)

typedef long long LL;

const LL maxN=1e5+7,MOD=1e9+9,INF=1e9+7,N=1007,sqr=383008016;

pii res[10];

LL n,m;

set<pii>q;

vector<pii>p;

long double ans,tmp;

long double dis(pii x, pii y)

{

long double res=((x.X-y.X)\*(x.X-y.X))+((x.Y-y.Y)\*(x.Y-y.Y));

return sqrt(res);

}

void add(LL x, LL y)

{

if(x>=0 && x<=n && y>=0 && y<=m)

q.insert(pii(x,y));

}

main()

{

cin>>n>>m;

add(0,0);add(n,0);

add(0,1);add(n-1,0);

add(1,0);add(n,1);

add(1,1);add(n-1,1);

add(0,m);add(n,m);

add(0,m-1);add(n-1,m);

add(1,m);add(n,m-1);

add(1,m-1);add(n-1,m-1);

for(auto it=q.begin();it!=q.end();it++)

p.pb(\*it);

rep(i,p.size())

{

rep(j,p.size())

{

rep(x,p.size())

{

rep(y,p.size())

{

if(i!=j && i!=x && i!=y && j!=x && j!=y && x!=y)

{

tmp=dis(p[i],p[j])+dis(p[j],p[x])+dis(p[x],p[y]);

if(tmp>=ans)

{

ans=tmp;

res[1]=p[i];

res[2]=p[j];

res[3]=p[x];

res[4]=p[y];

}

}

}

}

}

}

FOR(i,1,4)

cout<<res[i].X<<" "<<res[i].Y<<'\n';

return cout<<"\n",0;

}

<http://codeforces.com/contest/462/problem/B>

B. Appleman and Card Game

time limit per test

1 second

memory limit per test

256 megabytes

input

standard input

output

standard output

Appleman has *n* cards. Each card has an uppercase letter written on it. Toastman must choose *k* cards from Appleman's cards. Then Appleman should give Toastman some coins depending on the chosen cards. Formally, for each Toastman's card *i* you should calculate how much Toastman's cards have the letter equal to letter on *i*th, then sum up all these quantities, such a number of coins Appleman should give to Toastman.

Given the description of Appleman's cards. What is the maximum number of coins Toastman can get?

**Input**

The first line contains two integers *n* and *k* (1 ≤ *k* ≤ *n* ≤ 105). The next line contains *n* uppercase letters without spaces — the *i*-th letter describes the *i*-th card of the Appleman.

**Output**

Print a single integer – the answer to the problem.

**Examples**

**input**

15 10  
DZFDFZDFDDDDDDF

**output**

82

**input**

6 4  
YJSNPI

**output**

4

**Note**

In the first test example Toastman can choose nine cards with letter D and one additional card with any letter. For each card with D he will get 9 coins and for the additional card he will get 1 coin.

#include <bits/stdc++.h>

using namespace std;

typedef long long ll;

int n,k,a[26];

char s[100005];

ll ans;

int main(int argc, const char \* argv[]) {

#ifndef ONLINE\_JUDGE

freopen("input.in", "r", stdin);

#endif

scanf("%d%d",&n,&k);

scanf("%s",s);

for(int i = 0; s[i]; ++i) ++a[s[i]-'A'];

sort(a,a+26);

for(int i = 25; i>=0; --i)

ans += (ll) min(k,a[i])\*min(k,a[i]), k -= min(k,a[i]);

printf("%lld\n",ans);

return 0;

}

<http://codeforces.com/contest/237/problem/A>

A. Free Cash

time limit per test

2 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

Valera runs a 24/7 fast food cafe. He magically learned that next day *n* people will visit his cafe. For each person we know the arrival time: the *i*-th person comes exactly at *hi* hours *mi* minutes. The cafe spends less than a minute to serve each client, but if a client comes in and sees that there is no free cash, than he doesn't want to wait and leaves the cafe immediately.

Valera is very greedy, so he wants to serve all *n* customers next day (and get more profit). However, for that he needs to ensure that at each moment of time the number of working cashes is no less than the number of clients in the cafe.

Help Valera count the minimum number of cashes to work at his cafe next day, so that they can serve all visitors.

**Input**

The first line contains a single integer *n* (1 ≤ *n* ≤ 105), that is the number of cafe visitors.

Each of the following *n* lines has two space-separated integers *hi* and *mi* (0 ≤ *hi* ≤ 23; 0 ≤ *mi* ≤ 59), representing the time when the *i*-th person comes into the cafe.

Note that the time is given in the **chronological** order. All time is given within one 24-hour period.

**Output**

Print a single integer — the minimum number of cashes, needed to serve all clients next day.

**Examples**

**input**

4  
8 0  
8 10  
8 10  
8 45

**output**

2

**input**

3  
0 12  
10 11  
22 22

**output**

1

**Note**

In the first sample it is not enough one cash to serve all clients, because two visitors will come into cafe in 8:10. Therefore, if there will be one cash in cafe, then one customer will be served by it, and another one will not wait and will go away.

In the second sample all visitors will come in different times, so it will be enough one cash.

#include <iostream>

#include <cmath>

using namespace std;

int main(){

long int n,tt = 0;

cin >> n;

int h[n],m[n];

for(int i = 0;i < n;i++){

cin >> h[i] >> m[i];

}

long int t[n];

for(int i = 0;i < n;i++){

t[i] = 1;

}

if(n > 1 && h[0] == h[1] && m[0] == m[1]){

t[0]++;

}

for(int i = 1;i < n - 1;i++){

if(h[i] == h[i + 1] && m[i] == m[i + 1]){

if(h[i] != h[i - 1] || m[i] != m[i - 1]){

tt++;

}

t[tt]++;

}

}

long int mi = 1;

for(int i = 0;i < tt + 1;i++){

mi = max(mi ,t[i]);

}

cout << mi;

return 0;

}

<http://codeforces.com/contest/271/problem/B>

B. Prime Matrix

time limit per test

2 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

You've got an *n* × *m* matrix. The matrix consists of integers. In one move, you can apply a single transformation to the matrix: choose an arbitrary element of the matrix and increase it by 1. Each element can be increased an arbitrary number of times.

You are really curious about prime numbers. Let us remind you that a *prime number* is a positive integer that has exactly two distinct positive integer divisors: itself and number one. For example, numbers 2, 3, 5 are prime and numbers 1, 4, 6 are not.

A matrix is *prime* if at least one of the two following conditions fulfills:

* the matrix has a row with prime numbers only;
* the matrix has a column with prime numbers only;

Your task is to count the minimum number of moves needed to get a prime matrix from the one you've got.

**Input**

The first line contains two integers *n*, *m* (1 ≤ *n*, *m* ≤ 500) — the number of rows and columns in the matrix, correspondingly.

Each of the following *n* lines contains *m* integers — the initial matrix. All matrix elements are positive integers. All numbers in the initial matrix do not exceed 105.

The numbers in the lines are separated by single spaces.

**Output**

Print a single integer — the minimum number of moves needed to get a prime matrix from the one you've got. If you've got a prime matrix, print 0.

**Examples**

**input**

3 3  
1 2 3  
5 6 1  
4 4 1

**output**

1

**input**

2 3  
4 8 8  
9 2 9

**output**

3

**input**

2 2  
1 3  
4 2

**output**

0

**Note**

In the first sample you need to increase number 1 in cell (1, 1). Thus, the first row will consist of prime numbers: 2, 2, 3.

In the second sample you need to increase number 8 in cell (1, 2) three times. Thus, the second column will consist of prime numbers: 11, 2.

In the third sample you don't have to do anything as the second column already consists of prime numbers: 3, 2.

#include <iostream>

#include<vector>

#include <algorithm>

using namespace std;

int findNearestPrime(int num,std::vector<bool> primos){

int index = num;

while(primos[index]==false){

index+=1;

}

return (index-num);

}

std::vector<bool> getPrimes(){

std::vector<bool> primos (100053, true);

primos[0] = false;

primos[1] = false;

for (int i = 2; i<100053; i+=1){

if (primos[i]){

for (int j = i+i; j<100051; j+=i){

primos[j] = false;

};

}

};

return primos;

}

int main() {

std::vector<bool> primes = getPrimes();

int entradas[2];

int tmp;

int N = 0;

while(N<2 && scanf( "%d", &tmp) != EOF ) {

entradas[N++] = tmp;

}

std::vector<int> rows (entradas[0], 0);

std::vector<int> columns (entradas[1], 0);

std::vector<int> rowsAndColumns;

for (int i = 0; i<entradas[0];i+=1){

int row[entradas[1]];

int tmp;

int N = 0;

while(N<entradas[1] && scanf( "%d", &tmp) != EOF ) {

row[N++] = tmp;

}

for (int j = 0;j<entradas[1];j+=1){

if(primes[row[j]]==false){

rows[i]+=findNearestPrime(row[j],primes);

columns[j]+=findNearestPrime(row[j],primes);

}

}

}

rowsAndColumns.reserve( rows.size() + columns.size() ); // preallocate memory

rowsAndColumns.insert( rowsAndColumns.end(), rows.begin(), rows.end() );

rowsAndColumns.insert( rowsAndColumns.end(), columns.begin(), columns.end() );

std::sort(rowsAndColumns.begin(), rowsAndColumns.end());

std::cout << rowsAndColumns[0] << std::endl;

}

<http://codeforces.com/contest/49/problem/A>

A. Sleuth

time limit per test

2 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

Vasya plays the sleuth with his friends. The rules of the game are as follows: those who play for the first time, that is Vasya is the sleuth, he should investigate a "crime" and find out what is happening. He can ask any questions whatsoever that can be answered with "Yes" or "No". All the rest agree beforehand to answer the questions like that: if the question’s last letter is a vowel, they answer "Yes" and if the last letter is a consonant, they answer "No". Of course, the sleuth knows nothing about it and his task is to understand that.

Unfortunately, Vasya is not very smart. After 5 hours of endless stupid questions everybody except Vasya got bored. That’s why Vasya’s friends ask you to write a program that would give answers instead of them.

The English alphabet vowels are: A, E, I, O, U, Y

The English alphabet consonants are: B, C, D, F, G, H, J, K, L, M, N, P, Q, R, S, T, V, W, X, Z

**Input**

The single line contains a question represented by a non-empty line consisting of large and small Latin letters, spaces and a question mark. The line length does not exceed 100. It is guaranteed that the question mark occurs exactly once in the line — as the last symbol and that the line contains at least one letter.

**Output**

Print answer for the question in a single line: YES if the answer is "Yes", NO if the answer is "No".

Remember that in the reply to the question the last **letter**, not the last character counts. I. e. the spaces and the question mark do not count as letters.

**Examples**

**input**

Is it a melon?

**output**

NO

**input**

Is it an apple?

**output**

YES

**input**

Is it a banana ?

**output**

YES

**input**

Is it an apple and a banana simultaneouSLY?

**output**

YES

#include<stdio.h>

#include<string.h>

int main()

{

int l,i,p;

char s[100];

gets(s);

l=strlen(s);

p=l-2;

while(s[p]==32)

{

p--;

}

if(s[p]=='a'||s[p]=='a'-32||s[p]=='e'||s[p]=='e'-32||s[p]=='i'||s[p]=='i'-32||s[p]=='o'||s[p]=='o'-32||s[p]=='u'||s[p]=='u'-32||s[p]=='y'||s[p]=='y'-32)

{

printf("YES");

}

else

{

printf("NO");

}

return 0;

}

<http://codeforces.com/contest/369/problem/C>

C. Valera and Elections

time limit per test

1 second

memory limit per test

256 megabytes

input

standard input

output

standard output

The city Valera lives in is going to hold elections to the city Parliament.

The city has *n* districts and *n* - 1 bidirectional roads. We know that from any district there is a path along the roads to any other district. Let's enumerate all districts in some way by integers from 1 to *n*, inclusive. Furthermore, for each road the residents decided if it is the problem road or not. *A problem road* is a road that needs to be repaired.

There are *n* candidates running the elections. Let's enumerate all candidates in some way by integers from 1 to *n*, inclusive. If the candidate number *i* will be elected in the city Parliament, he will perform exactly one promise — to repair all problem roads on the way from the *i*-th district to the district 1, where the city Parliament is located.

Help Valera and determine the subset of candidates such that if all candidates from the subset will be elected to the city Parliament, all problem roads in the city will be repaired. If there are several such subsets, you should choose the subset consisting of the minimum number of candidates.

**Input**

The first line contains a single integer *n* (2 ≤ *n* ≤ 105) — the number of districts in the city.

Then *n* - 1 lines follow. Each line contains the description of a city road as three positive integers *xi*, *yi*, *ti* (1 ≤ *xi*, *yi* ≤ *n*, 1 ≤ *ti* ≤ 2) — the districts connected by the *i*-th bidirectional road and the road type. If *ti* equals to one, then the *i*-th road isn't the problem road; if *ti*equals to two, then the *i*-th road is the problem road.

It's guaranteed that the graph structure of the city is a tree.

**Output**

In the first line print a single non-negative number *k* — the minimum size of the required subset of candidates. Then on the second line print *k* space-separated integers *a*1, *a*2, ... *ak* — the numbers of the candidates that form the required subset. If there are multiple solutions, you are allowed to print any of them.

**Examples**

**input**

5  
1 2 2  
2 3 2  
3 4 2  
4 5 2

**output**

1  
5

**input**

5  
1 2 1  
2 3 2  
2 4 1  
4 5 1

**output**

1  
3

**input**

5  
1 2 2  
1 3 2  
1 4 2  
1 5 2

**output**

4  
5 4 3 2

#include <bits/stdc++.h>

using namespace std;

typedef long long ll;

typedef pair<int, int> ii;

int MOD = 1e9 + 7; ll INF = 1e18;

int n;

vector<int> g[200005], g2[200005], out;

int dp[200005];

void dfs(int u, int p = -1){

dp[u] = 0;

for(int v: g[u]) if(v - p){

dfs(v, u);

dp[u] |= dp[v];

}

for(int v: g2[u]) if(v - p){

dfs(v, u);

if(!dp[v]){

out.push\_back(v);

dp[v] = 1;

}

dp[u] |= dp[v];

}

}

int solve(){

cin >> n;

for(int i=0;i<n-1;i++){

int u, v, l; cin >> u >> v >> l; u--, v--, l--;

if(l & 1)

g2[u].push\_back(v), g2[v].push\_back(u);

else

g[u].push\_back(v), g[v].push\_back(u);

}

dfs(0);

cout << out.size() << endl;

for(int i=0;i<out.size();i++) cout << out[i] + 1 << " \n"[i==out.size()-1];

return 0;

}

int main(){

ios::sync\_with\_stdio(0);

// int t; cin >> t; while(t--)

solve();

// cout << (solve() ? "YES" : "NO") << endl;

return 0;

}

<http://codeforces.com/contest/50/problem/A>

A. Domino piling

time limit per test

2 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

You are given a rectangular board of *M* × *N* squares. Also you are given an unlimited number of standard domino pieces of 2 × 1squares. You are allowed to rotate the pieces. You are asked to place as many dominoes as possible on the board so as to meet the following conditions:

1. Each domino completely covers two squares.

2. No two dominoes overlap.

3. Each domino lies entirely inside the board. It is allowed to touch the edges of the board.

Find the maximum number of dominoes, which can be placed under these restrictions.

**Input**

In a single line you are given two integers *M* and *N* — board sizes in squares (1 ≤ *M* ≤ *N* ≤ 16).

**Output**

Output one number — the maximal number of dominoes, which can be placed.

**Examples**

**input**

2 4

**output**

4

**input**

3 3

**output**

4

#include<iostream>

using namespace std;

int main()

{

int M,N;

cin>>M>>N;

cout<<(M\*N)/2;

return 0;

}

<http://codeforces.com/contest/752/problem/F>

F. Santa Clauses and a Soccer Championship

time limit per test

2 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

The country Treeland consists of *n* cities connected with *n* - 1 bidirectional roads in such a way that it's possible to reach every city starting from any other city using these roads. There will be a soccer championship next year, and all participants are Santa Clauses. There are exactly 2*k* teams from 2*k* different cities.

During the first stage all teams are divided into *k* pairs. Teams of each pair play two games against each other: one in the hometown of the first team, and the other in the hometown of the other team. Thus, each of the 2*k* cities holds exactly one soccer game. However, it's not decided yet how to divide teams into pairs.

It's also necessary to choose several cities to settle players in. Organizers tend to use **as few cities as possible** to settle the teams.

Nobody wants to travel too much during the championship, so if a team plays in cities *u* and *v*, it wants to live in one of the cities on the shortest path between *u* and *v* (maybe, in *u* or in *v*). There is another constraint also: the teams from one pair must live in the same city.

Summarizing, the organizers want to divide 2*k* teams into pairs and settle them in the minimum possible number of cities *m* in such a way that teams from each pair live in the same city which lies between their hometowns.

**Input**

The first line of input contains two integers *n* and *k* (2 ≤ *n* ≤ 2·105, 2 ≤ 2*k* ≤ *n*) — the number of cities in Treeland and the number of pairs of teams, respectively.

The following *n* - 1 lines describe roads in Treeland: each of these lines contains two integers *a* and *b* (1 ≤ *a*, *b* ≤ *n*, *a* ≠ *b*) which mean that there is a road between cities *a* and *b*. It's guaranteed that there is a path between any two cities.

The last line contains 2*k* distinct integers *c*1, *c*2, ..., *c*2*k* (1 ≤ *ci* ≤ *n*), where *ci* is the hometown of the *i*-th team. All these numbers are distinct.

**Output**

The first line of output must contain the only positive integer *m* which should be equal to the minimum possible number of cities the teams can be settled in.

The second line should contain *m* distinct numbers *d*1, *d*2, ..., *dm* (1 ≤ *di* ≤ *n*) denoting the indices of the cities where the teams should be settled.

The *k* lines should follow, the *j*-th of them should contain 3 integers *uj*, *vj* and *xj*, where *uj* and *vj* are the hometowns of the *j*-th pair's teams, and *xj* is the city they should live in during the tournament. Each of the numbers *c*1, *c*2, ..., *c*2*k* should occur in all *uj*'s and *vj*'s exactly once. Each of the numbers *xj* should belong to {*d*1, *d*2, ..., *dm*}.

If there are several possible answers, print any of them.

**Example**

**input**

6 2  
1 2  
1 3  
2 4  
2 5  
3 6  
2 5 4 6

**output**

1  
2  
5 4 2  
6 2 2

**Note**

In the first test the orginizers can settle all the teams in the city number 2. The way to divide all teams into pairs is not important, since all requirements are satisfied anyway, because the city 2 lies on the shortest path between every two cities from {2, 4, 5, 6}.

#include <bits/stdc++.h>

#define F first

#define S second

#define pb push\_back

#define ld double

#define ll long long

#define int long long

using namespace std;

const int MAXN = 201 \* 1001 ;

int n , k ;

vector < int > adj[MAXN] ;

int sz[MAXN] , mark[MAXN];

void dfs1(int u , int p )

{

sz[u] = mark[u] ;

for(auto v : adj[u])

if(v!=p)

dfs1(v,u) , sz[u] += sz[v];

}

vector < int > v ;

void dfs2(int u , int p)

{

if(mark[u])

v.pb(u);

for(auto v :adj[u])

if(v!=p)

dfs2(v,u) ;

}

int32\_t main()

{

ios::sync\_with\_stdio(0);cin.tie(0);

int n,k;

cin >> n >> k ;

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

{

int x,y ;

cin >> x >> y ;

adj[x].pb(y);

adj[y].pb(x);

}

int x ;

for(int i = 0 ; i < 2 \* k ; i ++ )

cin >> x , mark[x] = 1 ;

dfs1(1,0);

int u = 1 , p = 0 ;

while(sz[u]>k)

{

bool f = 0 ;

for(auto v :adj[u] )

{

if(sz[v]>k&&v!=p)

{

f = 1 ;

p = u ;

u = v ;

break ;

}

}

if(!f)

break ;

}

cout<<1<<'\n'<<u<<'\n';

dfs2(u,0);

for(int i = 0 ; i < k ; i ++ )

cout<<v[i]<<' '<<v[i+k]<<' '<<u<<'\n';

}

<http://codeforces.com/contest/279/problem/B>

B. Books

time limit per test

2 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

When Valera has got some free time, he goes to the library to read some books. Today he's got *t* free minutes to read. That's why Valera took *n* books in the library and for each book he estimated the time he is going to need to read it. Let's number the books by integers from 1 to *n*. Valera needs *ai* minutes to read the *i*-th book.

Valera decided to choose an arbitrary book with number *i* and read the books one by one, starting from this book. In other words, he will first read book number *i*, then book number *i* + 1, then book number *i* + 2 and so on. He continues the process until he either runs out of the free time or finishes reading the *n*-th book. Valera reads each book up to the end, that is, he doesn't start reading the book if he doesn't have enough free time to finish reading it.

Print the maximum number of books Valera can read.

**Input**

The first line contains two integers *n* and *t* (1 ≤ *n* ≤ 105; 1 ≤ *t* ≤ 109) — the number of books and the number of free minutes Valera's got. The second line contains a sequence of *n* integers *a*1, *a*2, ..., *an* (1 ≤ *ai* ≤ 104), where number *ai* shows the number of minutes that the boy needs to read the *i*-th book.

**Output**

Print a single integer — the maximum number of books Valera can read.

**Examples**

**input**

4 5  
3 1 2 1

**output**

3

**input**

3 3  
2 2 3

**output**

1

#include <bits/stdc++.h>

#define eps 1e-8

#define inf 0x3f3f3f3f

#define INF 2e18

#define LL long long

#define ULL unsigned long long

#define PI acos(-1.0)

#define pb push\_back

#define mk make\_pair

#define pii pair<int,int>

#define pLL pair<LL,LL>

#define ff first

#define ss second

#define all(a) a.begin(),a.end()

#define SQR(a) ((a)\*(a))

#define Unique(a) sort(all(a)),a.erase(unique(all(a)),a.end())

#define min3(a,b,c) min(a,min(b,c))

#define max3(a,b,c) max(a,max(b,c))

#define min4(a,b,c,d) min(min(a,b),min(c,d))

#define max4(a,b,c,d) max(max(a,b),max(c,d))

#define print freopen("out.txt","w",stdout);

int left(int n)

{

return n<<1;

}

int right(int n)

{

return (n<<1)+1;

}

namespace patch

{

template<typename T>std::string to\_string(const T&n)

{

std::ostringstream stm;

stm<<n;

return stm.str();

}

}

using namespace std;

int Set(int N,int pos)

{

return N=N | (1<<pos);

}

int reset(int N,int pos)

{

return N= N & ~(1<<pos);

}

bool check(int N,int pos)

{

return (bool)(N & (1<<pos));

}

int ii[]= {-2,-2,-3,-1,-1,+1};

int jj[]= {+1,-1,-1,-2,-3,-2};

template <typename T> inline T GCD (T a,T b )

{

a = abs(a);

b = abs(b);

while ( b )

{

a = a % b;

swap ( a, b );

}

return a;

}

template <typename T> inline T LCM(T x,T y)

{

T tp = GCD(x,y);

if( (x / tp) \* 1. \* y > 9e18) return 9e18;

return (x / tp) \* y;

}

template <typename T> inline T BigMod(T A,T B,T M)

{

T ret = 1;

while(B)

{

if(B & 1) ret = (ret \* A) % M;

A = (A \* A) % M;

B = B >> 1;

}

return ret;

}

template <typename T> inline T ModInv (T A,T M)

{

return BigMod(A,M-2,M);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* End of template \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int a[100010],b[100010],c[100010];

int main()

{

int n,i,j,mx,x=0;

int temp=0;

long long int t,sum=0;

cin>>n>>t;

for (i = 0; i < n; ++i)

{

scanf("%d",&a[i]);

}

while(sum+a[temp]<=t && temp<n){

sum=sum+a[temp];

temp++;

x++;

}

mx=temp;

for(i=0;i<n;i++){

j=0;

sum=sum-a[i];x--;

while(sum+a[temp+j]<=t && temp+j<n){

sum=sum+a[temp+j];

j++;

x++;

}

if(x>mx)

mx=x;

temp=temp+j;

}

cout<<mx;

return 0;

}

<http://codeforces.com/contest/452/problem/F>

F. Permutation

time limit per test

1 second

memory limit per test

256 megabytes

input

standard input

output

standard output

You are given a permutation of numbers from 1 to *n*. Determine whether there's a pair of integers *a*, *b* (1 ≤ *a*, *b* ≤ *n*; *a* ≠ *b*) such that the element http://codeforces.com/predownloaded/b8/93/b8932c5b292015c8ebd81b9cee56747f5c542275.png (note, that it is usual division, not integer one) is between *a* and *b* in this permutation.

**Input**

First line consists of a single integer *n* (1 ≤ *n* ≤ 300000) — the size of permutation.

Second line contains *n* integers — the permutation itself.

**Output**

Print "YES", if such a pair exists, "NO" otherwise (in both cases without quotes, the answer is case insensitive).

**Examples**

**input**

4  
1 3 4 2

**output**

NO

**input**

5  
1 5 2 4 3

**output**

YES

**Note**

In the second example 2 is between 1 and 3. Additionally 4 is between 3 and 5.

#include <cstdio>

#include <cstring>

#include <algorithm>

using namespace std;

const int MAXN = 300005;

const int BASE = 10007;

const int MOD = 1000000007;

int pot[MAXN];

int inv[MAXN];

inline int add(int x, int y) {

x += y;

if (x >= MOD)

x -= MOD;

else if (x < 0)

x += MOD;

return x;

}

inline int mul(int x, int y) {

return (long long)x \* y % MOD;

}

int expo(int x, int y) {

int res = 1;

for (; y; y /= 2) {

if (y % 2)

res = mul(res, x);

x = mul(x, x);

}

return res;

}

struct fenwick {

int f[MAXN];

fenwick() {

memset(f, 0, sizeof f);

}

void update(int x, int val) {

for (; x < MAXN; x += x & -x)

f[x] = add(f[x], val);

}

int get(int x) {

int res = 0;

for (; x; x -= x & -x)

res = add(res, f[x]);

return res;

}

int query(int lo, int hi) {

return mul(pot[hi], add(get(hi), -get(lo - 1)));

}

};

int N;

int p[MAXN];

fenwick hsh, rev;

void load() {

scanf("%d", &N);

for (int i = 1; i <= N; i++)

scanf("%d", p + i);

}

bool solve() {

pot[0] = 1;

for (int i = 1; i <= N; i++) {

pot[i] = mul(pot[i - 1], BASE);

inv[i] = expo(pot[i], MOD - 2);

}

for (int i = 1; i <= N; i++) {

hsh.update(p[i], inv[p[i]]);

rev.update(N - p[i] + 1, inv[N - p[i] + 1]);

int len = min(p[i] - 1, N - p[i]);

if (hsh.query(p[i] - len, p[i] + len) != rev.query(N - p[i] + 1 - len, N - p[i] + 1 + len))

return true;

}

return false;

}

int main() {

load();

puts(solve() ? "YES": "NO");

return 0;

}

<http://codeforces.com/contest/550/problem/C>

C. Divisibility by Eight

time limit per test

2 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

You are given a non-negative integer *n*, its decimal representation consists of at most 100 digits and doesn't contain leading zeroes.

Your task is to determine if it is possible in this case to remove some of the digits (possibly not remove any digit at all) so that the result contains at least one digit, forms a non-negative integer, doesn't have leading zeroes and is divisible by 8. After the removing, it is forbidden to rearrange the digits.

If a solution exists, you should print it.

**Input**

The single line of the input contains a non-negative integer *n*. The representation of number *n* doesn't contain any leading zeroes and its length doesn't exceed 100 digits.

**Output**

Print "NO" (without quotes), if there is no such way to remove some digits from number *n*.

Otherwise, print "YES" in the first line and the resulting number after removing digits from number *n* in the second line. The printed number must be divisible by 8.

If there are multiple possible answers, you may print any of them.

**Examples**

**input**

3454

**output**

YES  
344

**input**

10

**output**

YES  
0

**input**

111111

**output**

NO

#include <bits/stdc++.h>

using namespace std;

typedef long long int ll;

vector <int> N;

int axe(int a, string B)

{

int n = B.size();

int x,i;

x= a%10;

for(i=n-1;i>=0;i--)

{

if(B[i]==(x+'0'))

{

a=a/10;

if(a==0) return 1;

else x=a%10;

}

}

return 0;

}

int main()

{

string A;

int i, flag = 0;

for(i=0;i\*8<=1000;i++)

{

N.push\_back(i\*8);

}

cin>>A;

for(i=0;i<N.size();i++)

{

if(axe(N[i],A))

{

cout<<"YES"<<endl;

cout<<N[i]<<endl;

flag=1;

break;

}

}

if(flag==0) cout<<"NO"<<endl;

return 0;

}

<http://codeforces.com/contest/691/problem/E>

E. Xor-sequences

time limit per test

3 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

You are given *n* integers *a*1,  *a*2,  ...,  *an*.

A sequence of integers *x*1,  *x*2,  ...,  *xk* is called a "xor-sequence" if for every 1  ≤  *i*  ≤  *k* - 1 the number of ones in the binary representation of the number *xi* http://codeforces.com/predownloaded/c1/5d/c15d2b716fe31376e2f869457fe96754821bf274.png *xi*+  1's is a multiple of 3 and http://codeforces.com/predownloaded/87/3b/873bfd9b14949e68c62e8e9c808646fcb76ab992.png for all 1 ≤ *i* ≤ *k*. The symbol http://codeforces.com/predownloaded/c1/5d/c15d2b716fe31376e2f869457fe96754821bf274.png is used for the binary exclusive or operation.

How many "xor-sequences" of length *k* exist? Output the answer modulo 109 + 7.

**Note if *a* = [1, 1] and *k* = 1 then the answer is 2, because you should consider the ones from *a* as different.**

**Input**

The first line contains two integers *n* and *k* (1 ≤ *n* ≤ 100, 1 ≤ *k* ≤ 1018) — the number of given integers and the length of the "xor-sequences".

The second line contains *n* integers *ai* (0 ≤ *ai* ≤ 1018).

**Output**

Print the only integer *c* — the number of "xor-sequences" of length *k* modulo 109 + 7.

**Examples**

**input**

5 2  
15 1 2 4 8

**output**

13

**input**

5 1  
15 1 2 4 8

**output**

5

#include <bits/stdc++.h>

#define repn(i, n) for(int i=1;i<=n;i++)

using namespace std;

const int N = 107;

const int K = 107;

const int MOD = 1e9 + 7;

typedef long long ll;

struct Matrix{

ll mat[N][N];

int row, col;

};

Matrix multiply(Matrix A, Matrix B){

Matrix ret;

ret.row = A.row;

ret.col = B.col;

for(int i=1;i<=ret.row;i++){

for(int j=1;j<=ret.col;j++){

ret.mat[i][j] = 0;

for(int k=1;k<=A.col;k++){

ret.mat[i][j] += A.mat[i][k] \* B.mat[k][j];

ret.mat[i][j] %= MOD;

}

}

}

return ret;

}

Matrix expo(Matrix base, ll p){

if(p == 1) return base;

if(p & 1) return multiply(base, expo(base, p-1));

Matrix ret = expo(base, p/2);

return multiply(ret, ret);

}

int n;

ll k;

ll a[K];

int main(){

ios\_base::sync\_with\_stdio(false);

cin >> n >> k;

repn(i, n) cin >> a[i];

Matrix base;

base.row = n;

base.col = n;

repn(i, n){

repn(j, n){

ll Xor = a[i] ^ a[j];

if(\_\_builtin\_popcountll(Xor) % 3 == 0) base.mat[i][j] = 1;

else base.mat[i][j] = 0;

}

}

if(k == 1){

cout << n << endl;

return 0;

}

Matrix ret = expo(base, k-1);

ll ans = 0;

repn(i, n) repn(j, n) ans += ret.mat[i][j], ans %= MOD;

cout << ans << endl;

return 0;

}

<http://codeforces.com/contest/492/problem/A>

A. Vanya and Cubes

time limit per test

1 second

memory limit per test

256 megabytes

input

standard input

output

standard output

Vanya got *n* cubes. He decided to build a pyramid from them. Vanya wants to build the pyramid as follows: the top level of the pyramid must consist of 1 cube, the second level must consist of 1 + 2 = 3 cubes, the third level must have 1 + 2 + 3 = 6 cubes, and so on. Thus, the *i*-th level of the pyramid must have 1 + 2 + ... + (*i* - 1) + *i* cubes.

Vanya wants to know what is the maximum height of the pyramid that he can make using the given cubes.

**Input**

The first line contains integer *n* (1 ≤ *n* ≤ 104) — the number of cubes given to Vanya.

**Output**

Print the maximum possible height of the pyramid in the single line.

**Examples**

**input**

1

**output**

1

**input**

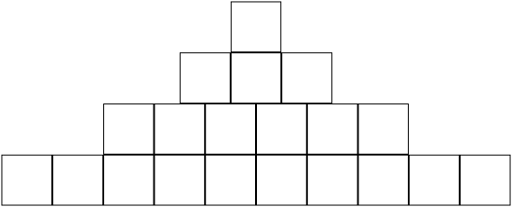
25

**output**

4

**Note**

Illustration to the second sample:



#if 1

#include <cstdlib>

#include <cstdio>

#include <cmath>

#include <algorithm>

#include <functional>

#include <cstring>

#include <string>

#include <vector>

#include <list>

#include <set>

#include <map>

#include <stack>

#include <queue>

#include <ctime>

#include <cassert>

#include <sstream>

#include <iostream>

#include <bitset>

#include <numeric>

using namespace std;

typedef long long LL;

typedef long double LD;

typedef pair<int , int> pii;

typedef vector <int> veci;

typedef vector <veci> graph;

const LD eps = 1e-9;

const LD pi = acos(-1.0);

const int inf = 1000 \* 1000 \* 1000;

const LL inf64 = LL(inf) \* inf;

#define mp make\_pair

#define pb push\_back

#define X first

#define Y second

#define iss istringstream

#define oss ostringstream

#define dbg(x) {cerr << #x << " = " << x << endl;}

#define dbgv(x) {cerr << #x << " ={"; for (int \_i = 0; \_i < x.size(); \_i++) {if (\_i) cerr << ", "; cerr << x[\_i];} cerr << "}" << endl;}

#define NAME "problem"

int main() {

//freopen("input.txt", "r", stdin); //freopen("output.txt", "w", stdout);

//freopen(NAME ".in","r",stdin);freopen(NAME ".out","w",stdout);

int s;

cin >> s;

int lvl = 0;

int last\_level = 0;

int cur = 0;

while (cur <= s) {

++lvl;

last\_level += lvl;

cur += last\_level;

}

--lvl;

cout << lvl << endl;

return 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#endif

<http://codeforces.com/contest/127/problem/A>

A. Wasted Time

time limit per test

2 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

Mr. Scrooge, a very busy man, decided to count the time he wastes on all sorts of useless stuff to evaluate the lost profit. He has already counted the time he wastes sleeping and eating. And now Mr. Scrooge wants to count the time he has wasted signing papers.

Mr. Scrooge's signature can be represented as a polyline *A*1*A*2... *An*. Scrooge signs like that: first it places a pen at the point *A*1, then draws a segment from point *A*1 to point *A*2, then he draws a segment from point *A*2 to point *A*3 and so on to point *An*, where he stops signing and takes the pen off the paper. At that the resulting line can intersect with itself and partially repeat itself but Scrooge pays no attention to it and never changes his signing style. As Scrooge makes the signature, he never takes the pen off the paper and his writing speed is constant — 50 millimeters per second.

Scrooge signed exactly *k* papers throughout his life and all those signatures look the same.

Find the total time Scrooge wasted signing the papers.

**Input**

The first line contains two integers *n* and *k* (2 ≤ *n* ≤ 100, 1 ≤ *k* ≤ 1000). Each of the following *n* lines contains the coordinates of the polyline's endpoints. The *i*-th one contains coordinates of the point *Ai* — integers *xi* and *yi*, separated by a space.

All points *Ai* are different. The absolute value of all coordinates does not exceed 20. The coordinates are measured in millimeters.

**Output**

Print one real number — the total time Scrooges wastes on signing the papers in seconds. The absolute or relative error should not exceed 10- 6.

**Examples**

**input**

2 1  
0 0  
10 0

**output**

0.200000000

**input**

5 10  
3 1  
-5 6  
-2 -1  
3 2  
10 0

**output**

6.032163204

**input**

6 10  
5 0  
4 0  
6 0  
3 0  
7 0  
2 0

**output**

3.000000000

#include<bits/stdc++.h>

using namespace std;

int main(){

int n,k;

scanf("%d %d",&n,&k);

long double ans = 0;

int lx,ly;

for(int i=0;i<n;i++){

int x,y;

scanf("%d %d",&x,&y);

if(i!=0){

ans += sqrt(1.0\*(x-lx)\*(x-lx) + (y-ly)\*(y-ly));

}

lx = x;

ly = y;

}

ans /= 50.0;

ans \*= 1.0\*k;

cout<<fixed;

cout<<setprecision(6);

cout<<ans<<endl;

return 0;

}

<http://codeforces.com/contest/686/problem/A>

A. Free Ice Cream

time limit per test

2 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

After their adventure with the magic mirror Kay and Gerda have returned home and sometimes give free ice cream to kids in the summer.

At the start of the day they have *x* ice cream packs. Since the ice cream is free, people start standing in the queue before Kay and Gerda's house even in the night. Each person in the queue wants either to take several ice cream packs for himself and his friends or to give several ice cream packs to Kay and Gerda (carriers that bring ice cream have to stand in the same queue).

If a carrier with *d* ice cream packs comes to the house, then Kay and Gerda take all his packs. If a child who wants to take *d* ice cream packs comes to the house, then Kay and Gerda will give him *d* packs if they have enough ice cream, otherwise the child will get no ice cream at all and will leave in distress.

Kay wants to find the amount of ice cream they will have after all people will leave from the queue, and Gerda wants to find the number of distressed kids.

**Input**

The first line contains two space-separated integers *n* and *x* (1 ≤ *n* ≤ 1000, 0 ≤ *x* ≤ 109).

Each of the next *n* lines contains a character '+' or '-', and an integer *di*, separated by a space (1 ≤ *di* ≤ 109). Record "+ *di*" in *i*-th line means that a carrier with *di* ice cream packs occupies *i*-th place from the start of the queue, and record "- *di*" means that a child who wants to take *di* packs stands in *i*-th place.

**Output**

Print two space-separated integers — number of ice cream packs left after all operations, and number of kids that left the house in distress.

**Examples**

**input**

5 7  
+ 5  
- 10  
- 20  
+ 40  
- 20

**output**

22 1

**input**

5 17  
- 16  
- 2  
- 98  
+ 100  
- 98

**output**

3 2

**Note**

Consider the first sample.

1. Initially Kay and Gerda have 7 packs of ice cream.
2. Carrier brings 5 more, so now they have 12 packs.
3. A kid asks for 10 packs and receives them. There are only 2 packs remaining.
4. Another kid asks for 20 packs. Kay and Gerda do not have them, so the kid goes away distressed.
5. Carrier bring 40 packs, now Kay and Gerda have 42 packs.
6. Kid asks for 20 packs and receives them. There are 22 packs remaining.

#include<iostream>

using namespace std;

int main()

{long long x,n,b,child=0;

cin>>n>>x;

long long ice =x;

char a;

for(int i=0;i<n;i++)

{

cin>>a;

cin>>b;

if(a=='+')

{

ice =ice+b;

}

else

{

if(ice>=b)

{

ice =ice-b;

}

else{child++;}

}

}

cout<<ice<<" "<<child;

return 0;

}

<http://codeforces.com/contest/313/problem/C>

C. Ilya and Matrix

time limit per test

1 second

memory limit per test

256 megabytes

input

standard input

output

standard output

Ilya is a very good-natured lion. He likes maths. Of all mathematical objects, his favourite one is matrices. Now he's faced a complicated matrix problem he needs to solve.

He's got a square 2*n* × 2*n*-sized matrix and 4*n* integers. You need to arrange all these numbers in the matrix (put each number in a single individual cell) so that the *beauty* of the resulting matrix with numbers is maximum.

The *beauty* of a 2*n* × 2*n*-sized matrix is an integer, obtained by the following algorithm:

1. Find the maximum element in the matrix. Let's denote it as *m*.
2. If *n* = 0, then the beauty of the matrix equals *m*. Otherwise, a matrix can be split into 4 non-intersecting 2*n*- 1 × 2*n*- 1-sized submatrices, then the beauty of the matrix equals the sum of number *m* and other four beauties of the described submatrices.

As you can see, the algorithm is recursive.

Help Ilya, solve the problem and print the resulting maximum beauty of the matrix.

**Input**

The first line contains integer 4*n* (1 ≤ 4*n* ≤ 2·106). The next line contains 4*n* integers *ai* (1 ≤ *ai* ≤ 109) — the numbers you need to arrange in the 2*n* × 2*n*-sized matrix.

**Output**

On a single line print the maximum value of the beauty of the described matrix.

Please, do not use the %lld specifier to read or write 64-bit integers in С++. It is preferred to use the cin, cout streams or the %I64dspecifier.

**Examples**

**input**

1  
13

**output**

13

**input**

4  
1 2 3 4

**output**

14

**Note**

Consider the second sample. You need to arrange the numbers in the matrix as follows:

1 2  
3 4

Then the beauty of the matrix will equal: 4 + 1 + 2 + 3 + 4 = 14.

#include <bits/stdc++.h>

using namespace std;

int main()

{

/\* #ifndef ONLINE\_JUDGE

freopen("input.in","r",stdin);

freopen("output.out","w",stdout);

#endif\*/

int n;

while(scanf("%d",&n)!=EOF)

{

long long ans=0;

vector<long long>vec;

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

{

long long x;

scanf("%lld",&x);

vec.push\_back(x);

}

sort(vec.rbegin(),vec.rend());

for(int i=1;i<=n;i\*=4)

ans+=accumulate(vec.begin(),vec.begin()+i,0LL);

printf("%lld\n",ans);

}

return 0;

}

<http://codeforces.com/contest/741/problem/A>

A. Arpa's loud Owf and Mehrdad's evil plan

time limit per test

1 second

memory limit per test

256 megabytes

input

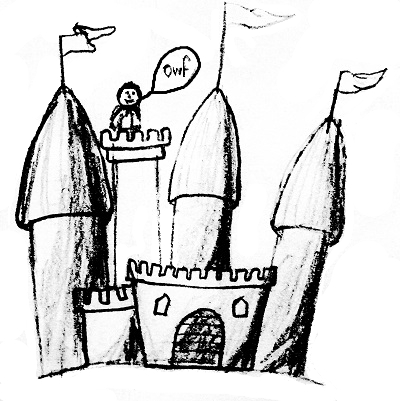
standard input

output

standard output

*As you have noticed, there are lovely girls in Arpa’s land.*

People in Arpa's land are numbered from 1 to *n*. Everyone has exactly one crush, *i*-th person's crush is person with the number *crushi*.



Someday Arpa shouted *Owf* loudly from the top of the palace and a funny game started in Arpa's land. The rules are as follows.

The game consists of rounds. Assume person *x* wants to start a round, he calls *crushx* and says: "Oww...wwf" (the letter w is repeated *t* times) and cuts off the phone immediately. If *t* > 1 then *crushx* calls *crushcrushx* and says: "Oww...wwf" (the letter w is repeated *t* - 1times) and cuts off the phone immediately. The round continues until some person receives an "Owf" (*t* = 1). This person is called the *Joon-Joon* of the round. There can't be two rounds at the same time.

Mehrdad has an evil plan to make the game more funny, he wants to find smallest *t* (*t* ≥ 1) such that for each person *x*, if *x* starts some round and *y* becomes the Joon-Joon of the round, then by starting from *y*, *x* would become the Joon-Joon of the round. Find such *t* for Mehrdad if it's possible.

Some strange fact in Arpa's land is that someone can be himself's crush (i.e. *crushi* = *i*).

**Input**

The first line of input contains integer *n* (1 ≤ *n* ≤ 100) — the number of people in Arpa's land.

The second line contains *n* integers, *i*-th of them is *crushi* (1 ≤ *crushi* ≤ *n*) — the number of *i*-th person's crush.

**Output**

If there is no *t* satisfying the condition, print -1. Otherwise print such smallest *t*.

**Examples**

**input**

4  
2 3 1 4

**output**

3

**input**

4  
4 4 4 4

**output**

-1

**input**

4  
2 1 4 3

**output**

1

**Note**

In the first sample suppose *t* = 3.

If the first person starts some round:

The first person calls the second person and says "Owwwf", then the second person calls the third person and says "Owwf", then the third person calls the first person and says "Owf", so the first person becomes Joon-Joon of the round. So the condition is satisfied if *x* is 1.

The process is similar for the second and the third person.

If the fourth person starts some round:

The fourth person calls himself and says "Owwwf", then he calls himself again and says "Owwf", then he calls himself for another time and says "Owf", so the fourth person becomes Joon-Joon of the round. So the condition is satisfied when *x* is 4.

In the last example if the first person starts a round, then the second person becomes the Joon-Joon, and vice versa.

#include<bits/stdc++.h>

#define MAXN 102

using namespace std;

int n, crush[MAXN], vst[MAXN];

int gcd(int a, int b){

if(b == 0)

return a;

return gcd(b, a%b);

}

int nwd(int x, int y){

return (x/gcd(x,y))\*y;

}

int main(){

scanf("%d", &n);

for(int i=0; i<n; ++i){

scanf("%d", &crush[i]);

crush[i]--;

}

int ans=1;

for(int i=0; i<n; ++i)

if( vst[i] == 0){

vst[i] =1;

int c = crush[i], len=1;

while( vst[c] == 0){

vst[c] =1;

len++;

c = crush[c];

}

//cout<<len<<endl;

if( c != i ) { printf("-1\n"); return 0;}

if( len %2 == 0) ans = nwd(ans,len/2);

else ans = nwd(ans, len);

}

printf("%d\n", ans);

return 0;

}

<http://codeforces.com/contest/14/problem/B>

B. Young Photographer

time limit per test

2 seconds

memory limit per test

64 megabytes

input

standard input

output

standard output

Among other things, Bob is keen on photography. Especially he likes to take pictures of sportsmen. That was the reason why he placed himself in position *x*0 of a long straight racetrack and got ready to take pictures. But the problem was that not all the runners passed him. The total amount of sportsmen, training at that racetrack, equals *n*. And each of them regularly runs distances within a particular segment of the racetrack, which is the same for each sportsman. For example, the first sportsman runs from position *a*1 to position *b*1, the second — from *a*2 to *b*2

What is the minimum distance that Bob should move to have a chance to take pictures of each sportsman? Bob can take a picture of a sportsman, if he stands within the segment that this sportsman covers on the racetrack.

**Input**

The first line of the input file contains integers *n* and *x*0 (1 ≤ *n* ≤ 100; 0 ≤ *x*0 ≤ 1000). The following *n* lines contain pairs of integers *ai*, *bi* (0 ≤ *ai*, *bi* ≤ 1000; *ai* ≠ *bi*).

**Output**

Output the required minimum distance in the same units as the positions on the racetrack. If there is no such a position, output -1.

**Examples**

**input**

3 3  
0 7  
14 2  
4 6

**output**

1

#include <iostream>

#include <algorithm>

using namespace std;

int main()

{

ios\_base::sync\_with\_stdio(false);

cin.tie(nullptr);

int n, x; cin >> n >> x;

int x1[n];

int x2[n];

for (int i = 0; i < n; i++)

{

cin >> x1[i] >> x2[i];

if (x1[i] > x2[i]) swap(x1[i], x2[i]);

}

sort(x1, x1 + n);

sort(x2, x2 + n);

int mn = x1[n - 1];

int mx = x2[0];

if (mn > mx) cout << "-1\n";

else if (x >= mn && x <= mx) cout << "0\n";

else cout << min(abs(x - mn), abs(x - mx)) << '\n';

return 0;

}

<http://codeforces.com/contest/670/problem/A>

A. Holidays

time limit per test

1 second

memory limit per test

256 megabytes

input

standard input

output

standard output

On the planet Mars a year lasts exactly *n* days (there are no leap years on Mars). But Martians have the same weeks as earthlings — 5 work days and then 2 days off. Your task is to determine the minimum possible and the maximum possible number of days off per year on Mars.

**Input**

The first line of the input contains a positive integer *n* (1 ≤ *n* ≤ 1 000 000) — the number of days in a year on Mars.

**Output**

Print two integers — the minimum possible and the maximum possible number of days off per year on Mars.

**Examples**

**input**

14

**output**

4 4

**input**

2

**output**

0 2

**Note**

In the first sample there are 14 days in a year on Mars, and therefore independently of the day a year starts with there will be exactly 4days off .

In the second sample there are only 2 days in a year on Mars, and they can both be either work days or days off.

#include <iostream>

using namespace std;

int main(){

long long a,b,c;

cin>>a;

b=(a/7)\*2;

c=b;

if(a%7==6)

{

c+=1;

}

if(a%7==1)

{

b+=1;

}

if(a%7>1)

{

b+=2;

}

cout<<c<<" "<<b;

return 0;

}

<http://codeforces.com/contest/204/problem/A>

A. Little Elephant and Interval

time limit per test

2 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

The Little Elephant very much loves sums on intervals.

This time he has a pair of integers *l* and *r* (*l* ≤ *r*). The Little Elephant has to find the number of such integers *x* (*l* ≤ *x* ≤ *r*), that the first digit of integer *x* equals the last one (in decimal notation). For example, such numbers as 101, 477474 or 9 will be included in the answer and 47, 253 or 1020 will not.

Help him and count the number of described numbers *x* for a given pair *l* and *r*.

**Input**

The single line contains a pair of integers *l* and *r* (1 ≤ *l* ≤ *r* ≤ 1018) — the boundaries of the interval.

Please, do not use the %lld specifier to read or write 64-bit integers in С++. It is preferred to use cin, cout streams or the %I64dspecifier.

**Output**

On a single line print a single integer — the answer to the problem.

**Examples**

**input**

2 47

**output**

12

**input**

47 1024

**output**

98

**Note**

In the first sample the answer includes integers 2, 3, 4, 5, 6, 7, 8, 9, 11, 22, 33, 44.

#include <bits/stdc++.h>

using namespace std;

typedef long long ll;

ll comp(ll numb){

ll ans = 0;

string str="";

ll temp=numb;

while(temp){

str+=(temp%10)+'0';

temp/=10;

}

for(int i = 0; i < str.length()/2;i++)swap(str[i],str[str.length()-i-1]);

// cout << numb << endl << str << endl;

if(numb<10)return numb;

for(int i = 1; i <= 9; i++){

for(int e = 1; e < str.length(); e++){

if(e == 1){

ans++;

continue;

}

if(e==2){

ans++;

continue;

}

ll init =0;

for(int f = 1; f<=(e-2); f++){

init=init\*10 + 9;

}

ans+=init+1;

}

}

string initStr=str;

//cout << ans << " cia : " << endl;

int nowLen = initStr.length();

ll su=numb;

int first = str[0]-'0';

while((su%10)!=first){

su--;

}

str="";

while(su){

str+= (su%10+'0');

su/=10;

}

for(int i = 0; i < str.length()/2;i++)swap(str[i],str[str.length()-1-i]);

if(str.length()<nowLen)return ans;

// cout << "HERE" << endl;

int lea=(str[0]-'0');

for(int i = 1; i<=lea; i++){

temp =numb;

while(temp%10 != i){

temp--;

}

str="";

while(temp){

str+=(temp%10 + '0');

temp/=10;

}

for(int l = 0; l < str.length()/2;l++)swap(str[l],str[str.length()-l-1]);

if(temp<0)continue;

if(str.length()!=nowLen)continue;

ll sia=0;

for(int e = 1; e <= str.length()-2; e++){

if(i==lea){

sia=sia\*10+(str[e]-'0');

}else sia=sia\*10+9;

}

//cout << sia << "<cia" << endl;

ans+=sia+1;

}

return ans;

}

int main(){

ll l,r;

cin >> l >> r;

cout << comp(r)-comp(l-1);

return 0;

}

<http://codeforces.com/contest/735/problem/B>

B. Urbanization

time limit per test

2 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

Local authorities have heard a lot about combinatorial abilities of Ostap Bender so they decided to ask his help in the question of urbanization. There are *n* people who plan to move to the cities. The wealth of the *i* of them is equal to *ai*. Authorities plan to build two cities, first for *n*1 people and second for *n*2 people. Of course, each of *n* candidates can settle in only one of the cities. Thus, first some subset of candidates of size *n*1 settle in the first city and then some subset of size *n*2 is chosen among the remaining candidates and the move to the second city. All other candidates receive an official refuse and go back home.

To make the statistic of local region look better in the eyes of their bosses, local authorities decided to pick subsets of candidates in such a way that **the sum of arithmetic mean** of wealth of people in each of the cities is as large as possible. Arithmetic mean of wealth in one city is the sum of wealth *ai* among all its residents divided by the number of them (*n*1 or *n*2 depending on the city). The division should be done in real numbers without any rounding.

Please, help authorities find the optimal way to pick residents for two cities.

**Input**

The first line of the input contains three integers *n*, *n*1 and *n*2 (1 ≤ *n*, *n*1, *n*2 ≤ 100 000, *n*1 + *n*2 ≤ *n*) — the number of candidates who want to move to the cities, the planned number of residents of the first city and the planned number of residents of the second city.

The second line contains *n* integers *a*1, *a*2, ..., *an* (1 ≤ *ai* ≤ 100 000), the *i*-th of them is equal to the wealth of the *i*-th candidate.

**Output**

Print one real value — the maximum possible sum of arithmetic means of wealth of cities' residents. You answer will be considered correct if its absolute or relative error does not exceed 10- 6.

Namely: let's assume that your answer is *a*, and the answer of the jury is *b*. The checker program will consider your answer correct, if http://codeforces.com/predownloaded/c6/2e/c62ea64d4651240724c5ac4779b671c741edec24.png.

**Examples**

**input**

2 1 1  
1 5

**output**

6.00000000

**input**

4 2 1  
1 4 2 3

**output**

6.50000000

**Note**

In the first sample, one of the optimal solutions is to move candidate 1 to the first city and candidate 2 to the second.

In the second sample, the optimal solution is to pick candidates 3 and 4 for the first city, and candidate 2 for the second one. Thus we obtain (*a*3 + *a*4) / 2 + *a*2 = (3 + 2) / 2 + 4 = 6.5

#include<iostream>

#include<cstdio>

#include<algorithm>

using namespace std;

int main()

{

double a[100010],s=0,t=0;int i,b,j,k,l,d,c;

scanf("%d %d %d",&b,&c,&d);

for(i=0;i<b;i++)

{

scanf("%lf",&a[i]);

}

sort(a,a+b);if(c<d){k=c;l=d;}else{k=d;l=c;}

for(i=b-1;i>=b-k;i--)s=s+a[i];

for(i=b-k-1;i>=b-k-l;i--)t=t+a[i];printf("%.8lf\n",(s/(double)k)+(t/(double)l));

}

<http://codeforces.com/contest/660/problem/C>

C. Hard Process

time limit per test

1 second

memory limit per test

256 megabytes

input

standard input

output

standard output

You are given an array *a* with *n* elements. Each element of *a* is either 0 or 1.

Let's denote the length of the longest subsegment of consecutive elements in *a*, consisting of only numbers one, as *f*(*a*). You can change no more than *k* zeroes to ones to maximize *f*(*a*).

**Input**

The first line contains two integers *n* and *k* (1 ≤ *n* ≤ 3·105, 0 ≤ *k* ≤ *n*) — the number of elements in *a* and the parameter *k*.

The second line contains *n* integers *ai* (0 ≤ *ai* ≤ 1) — the elements of *a*.

**Output**

On the first line print a non-negative integer *z* — the maximal value of *f*(*a*) after no more than *k* changes of zeroes to ones.

On the second line print *n* integers *aj* — the elements of the array *a* after the changes.

If there are multiple answers, you can print any one of them.

**Examples**

**input**

7 1  
1 0 0 1 1 0 1

**output**

4  
1 0 0 1 1 1 1

**input**

10 2  
1 0 0 1 0 1 0 1 0 1

**output**

5  
1 0 0 1 1 1 1 1 0 1

#include <bits/stdc++.h>

#define L(i, m, n) for(int i(m);i < n;i++)

#define pb push\_back

#define D(X) cout<<" "<<#X": "<<X<<endl;

#define in(x) cin >> x

#define clr(A, V) L(i, 0, sizeof(A)) A[i]=V

#define ff first

#define ss second

#define RF(X) freopen(X, "r", stdin)

#define WF(X) freopen(X, "w", stdout)

using namespace std;

typedef long long ll;

typedef pair<ll,ll> pll;

typedef vector<int> vi;

typedef pair<int,int> pii;

typedef vector<pii> vpii;

typedef pair<int, string> pis;

typedef vector<string> vs;

typedef pair<pair<int, int>, pair<int, int > > piiii;

const int mx = 300009;

int a[mx];

int n, k, p1, p2, c, ans, l, r,K;

int main(){

in(n), in(k);L(i, 0, n) in(a[i]);

K=k;

while(p1<n&&p2<n){

if(a[p2]||(!a[p2]&&k)){

c++, k-=(a[p2]?0:1),p2++;

if(c>ans)ans=c,l=p1,r=p2-1;

}

else{

if(c>ans)ans=c,l=p1,r=p2-1;

while(a[p1])c-=(c>0),p1++;

c--, p1++,k++;

}

}

cout<<ans<<endl;

L(i,0,n)

if(i>=l&&i<=r&&K)cout<<1<<" ";

else cout<<a[i]<<" ";

return 0;

}

<http://codeforces.com/contest/678/problem/C>

C. Joty and Chocolate

time limit per test

1 second

memory limit per test

256 megabytes

input

standard input

output

standard output

Little Joty has got a task to do. She has a line of *n* tiles indexed from 1 to *n*. She has to paint them in a strange pattern.

An unpainted tile should be painted Red if it's index is divisible by *a* and an unpainted tile should be painted Blue if it's index is divisible by *b*. So the tile with the number divisible by *a* and *b* can be either painted Red or Blue.

After her painting is done, she will get *p* chocolates for each tile that is painted Red and *q* chocolates for each tile that is painted Blue.

Note that she can paint tiles in any order she wants.

Given the required information, find the maximum number of chocolates Joty can get.

**Input**

The only line contains five integers *n*, *a*, *b*, *p* and *q* (1 ≤ *n*, *a*, *b*, *p*, *q* ≤ 109).

**Output**

Print the only integer *s* — the maximum number of chocolates Joty can get.

Note that the answer can be too large, so you should use 64-bit integer type to store it. In C++ you can use the long long integer type and in Java you can use long integer type.

**Examples**

**input**

5 2 3 12 15

**output**

39

**input**

20 2 3 3 5

**output**

51

#include <stdio.h>

#include <algorithm>

#define lli long long int

using namespace std;

int gcd(int a, int b) {

if (!b) {

return a;

}

return gcd(b, a % b);

}

lli lcm(int a, int b) {

return (1LL \* a / gcd(a, b)) \* b;

}

int main(void) {

int n, a, b, p, q;

scanf(" %d %d %d %d %d", &n, &a, &b, &p, &q);

lli mini = lcm(a, b);

lli x = (1LL \* n / a) \* p;

lli y = (1LL \* n / b) \* q;

lli z = (1LL \* n / mini) \* (min(p, q));

lli res = x + y - z;

printf("%I64d\n", res);

return 0;

}

<http://codeforces.com/contest/451/problem/D>

D. Count Good Substrings

time limit per test

2 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

We call a string *good*, if after merging all the consecutive equal characters, the resulting string is palindrome. For example, "aabba" is good, because after the merging step it will become "aba".

Given a string, you have to find two values:

1. the number of good substrings of even length;
2. the number of good substrings of odd length.

**Input**

The first line of the input contains a single string of length *n* (1 ≤ *n* ≤ 105). Each character of the string will be either 'a' or 'b'.

**Output**

Print two space-separated integers: the number of good substrings of even length and the number of good substrings of odd length.

**Examples**

**input**

bb

**output**

1 2

**input**

baab

**output**

2 4

**input**

babb

**output**

2 5

**input**

babaa

**output**

2 7

**Note**

In example 1, there are three good substrings ("b", "b", and "bb"). One of them has even length and two of them have odd length.

In example 2, there are six good substrings (i.e. "b", "a", "a", "b", "aa", "baab"). Two of them have even length and four of them have odd length.

In example 3, there are seven good substrings (i.e. "b", "a", "b", "b", "bb", "bab", "babb"). Two of them have even length and five of them have odd length.

*Definitions*

A substring *s*[*l*, *r*] (1 ≤ *l* ≤ *r* ≤ *n*) of string *s* = *s*1*s*2... *sn* is string *slsl*+ 1... *sr*.

A string *s* = *s*1*s*2... *sn* is a palindrome if it is equal to string *snsn*- 1... *s*1.

#include<bits/stdc++.h>

using namespace std;

long long int za,zb,oa,ob,ok,zk;

int main()

{

string s;

cin>>s;

for(int i=0 ; i<s.size() ; i++)

{

ok++;

if(s[i]=='a')

{

if(i%2==0)

{

ok+=za;

zk+=oa;

}

else

{

zk+=za;

ok+=oa;

}

}

if(s[i]=='b')

{

if(i%2==0)

{

ok+=zb;

zk+=ob;

}

else

{

zk+=zb;

ok+=ob;

}

}

if(i%2==1)

{

if(s[i]=='a')

{

oa++;

}

else

ob++;

}

if(i%2==0)

{

if(s[i]=='a')

{

za++;

}

else

{

zb++;

}

}

}

cout<<zk<<" "<<ok<<endl ;

}

<http://codeforces.com/contest/573/problem/A>

A. Bear and Poker

time limit per test

2 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

Limak is an old brown bear. He often plays poker with his friends. Today they went to a casino. There are *n* players (including Limak himself) and right now all of them have bids on the table. *i*-th of them has bid with size *ai* dollars.

Each player can double his bid any number of times and triple his bid any number of times. The casino has a great jackpot for making all bids equal. Is it possible that Limak and his friends will win a jackpot?

**Input**

First line of input contains an integer *n* (2 ≤ *n* ≤ 105), the number of players.

The second line contains *n* integer numbers *a*1, *a*2, ..., *an* (1 ≤ *ai* ≤ 109) — the bids of players.

**Output**

Print "Yes" (without the quotes) if players can make their bids become equal, or "No" otherwise.

**Examples**

**input**

4  
75 150 75 50

**output**

Yes

**input**

3  
100 150 250

**output**

No

**Note**

In the first sample test first and third players should double their bids twice, second player should double his bid once and fourth player should both double and triple his bid.

It can be shown that in the second sample test there is no way to make all bids equal.

#include<bits/stdc++.h>

using namespace std;

int main(){

int n;

long long int a[100000];

cin>>n;

for(int i=0;i<n;i++){

cin>>a[i];

while(a[i]%2==0){

a[i]/=2;

}

while(a[i]%3==0){

a[i]/=3;

}

}

long long int x=a[0];

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

if(x==a[i]) continue;

else {cout<<"No"<<endl; return 0;}

}

cout<<"Yes"<<endl;

}

<http://codeforces.com/contest/25/problem/A>

A. IQ test

time limit per test

2 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

Bob is preparing to pass IQ test. The most frequent task in this test is to find out which one of the given *n* numbers differs from the others. Bob observed that one number usually differs from the others in evenness. Help Bob — to check his answers, he needs a program that among the given *n* numbers finds one that is different in evenness.

**Input**

The first line contains integer *n* (3 ≤ *n* ≤ 100) — amount of numbers in the task. The second line contains *n* space-separated natural numbers, not exceeding 100. It is guaranteed, that exactly one of these numbers differs from the others in evenness.

**Output**

Output index of number that differs from the others in evenness. Numbers are numbered from 1 in the input order.

**Examples**

**input**

5  
2 4 7 8 10

**output**

3

**input**

4  
1 2 1 1

**output**

2

#include <iostream>

using namespace std;

int main()

{int n, s,k=0,b=0,h,z;

cin>>n;

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

//for (int j=0; j<=n;j++)

{cin>>s;

if(s%2==0){k++;h=i;}

else {b++;z=i;}

}

if (k==1 ){cout <<h<<" "<<endl;

return 0;

}

else cout <<z;

return 0;

}

<http://codeforces.com/contest/710/problem/A>

A. King Moves

time limit per test

1 second

memory limit per test

256 megabytes

input

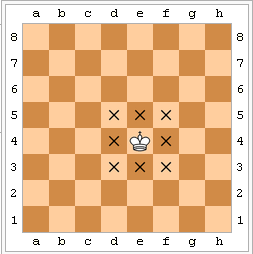
standard input

output

standard output

The only king stands on the standard chess board. You are given his position in format "cd", where *c* is the column from 'a' to 'h' and *d*is the row from '1' to '8'. Find the number of moves permitted for the king.

Check the king's moves here https://en.wikipedia.org/wiki/King\_(chess).

King moves from the position e4

**Input**

The only line contains the king's position in the format "cd", where 'c' is the column from 'a' to 'h' and 'd' is the row from '1' to '8'.

**Output**

Print the only integer *x* — the number of moves permitted for the king.

**Example**

**input**

e4

**output**

8

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<math.h>

int main()

{

int c=8;

char s[5];

scanf("%s",s);

if((s[0]=='a')||(s[0]=='h')||(s[1]=='1')||(s[1]=='8'))

c=5;

if((strcmp(s,"a1")==0)||(strcmp(s,"a8")==0)||(strcmp(s,"h1")==0)||(strcmp(s,"h8")==0))

c=3;

printf("%d",c);

return 0;

}

<http://codeforces.com/contest/2/problem/B>

B. The least round way

time limit per test

2 seconds

memory limit per test

64 megabytes

input

standard input

output

standard output

There is a square matrix *n* × *n*, consisting of non-negative integer numbers. You should find such a way on it that

* starts in the upper left cell of the matrix;
* each following cell is to the right or down from the current cell;
* the way ends in the bottom right cell.

Moreover, if we multiply together all the numbers along the way, the result should be the least "round". In other words, it should end in the least possible number of zeros.

**Input**

The first line contains an integer number *n* (2 ≤ *n* ≤ 1000), *n* is the size of the matrix. Then follow *n* lines containing the matrix elements (non-negative integer numbers not exceeding 109).

**Output**

In the first line print the least number of trailing zeros. In the second line print the correspondent way itself.

**Examples**

**input**

3  
1 2 3  
4 5 6  
7 8 9

**output**

0  
DDRR

#include<cstdio>

#include<cstring>

#include<algorithm>

#include<cctype>

#include<ctime>

#include<cstdlib>

#include<string>

#include<queue>

#define Rep(x,a,b) for (int x=a;x<=b;x++)

#define Per(x,a,b) for (int x=a;x>=b;x--)

#define ll long long

using namespace std;

inline int IN(){

int x=0,ch=getchar(),f=1;

while (!isdigit(ch)&&(ch!='-')&&(ch!=EOF)) ch=getchar();

if (ch=='-'){f=-1;ch=getchar();}

while (isdigit(ch)){x=(x<<1)+(x<<3)+ch-'0';ch=getchar();}

return x\*f;

}

bool hvz=0;

int poxz,n;

int a[1005][1005][2];

int dp[1005][1005][2];

void DP(int ct){

dp[1][1][ct]=a[1][1][ct];

Rep(i,2,n){

dp[1][i][ct]=dp[1][i-1][ct]+a[1][i][ct];

dp[i][1][ct]=dp[i-1][1][ct]+a[i][1][ct];

}

Rep(i,2,n) Rep(j,2,n)

dp[i][j][ct]=min(dp[i-1][j][ct],dp[i][j-1][ct])+a[i][j][ct];

}

void Print(int x,int y,int ct){

if (x==n&&y==n) printf("%d\n",dp[x][y][ct]);

if (x==1&&y==1) return;

if (x==1){

Print(x,y-1,ct);

putchar('R');

}

else if (y==1){

Print(x-1,y,ct);

putchar('D');

}

else{

if (dp[x][y-1][ct]+a[x][y][ct]==dp[x][y][ct]){

Print(x,y-1,ct);

putchar('R');

}

else{

Print(x-1,y,ct);

putchar('D');

}

}

}

int main(){

n=IN();

Rep(i,1,n) Rep(j,1,n){

int x=IN();

if (x==0){hvz=1;x=10;poxz=i;}

while (x%2==0) a[i][j][0]++,x/=2;

while (x%5==0) a[i][j][1]++,x/=5;

}

DP(0);DP(1);

if (hvz){

if (dp[n][n][0]==0) Print(n,n,0);

else if (dp[n][n][1]==0) Print(n,n,1);

else{

puts("1");

Rep(i,2,poxz) putchar('D');

Rep(i,2,n) putchar('R');

Rep(i,poxz+1,n) putchar('D');

}

}

else if (dp[n][n][0]<dp[n][n][1]) Print(n,n,0);

else Print(n,n,1);

}

<http://codeforces.com/contest/735/problem/A>

A. Ostap and Grasshopper

time limit per test

2 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

On the way to Rio de Janeiro Ostap kills time playing with a grasshopper he took with him in a special box. Ostap builds a line of length *n*such that some cells of this line are empty and some contain obstacles. Then, he places his grasshopper to one of the empty cells and a small insect in another empty cell. The grasshopper wants to eat the insect.

Ostap knows that grasshopper is able to jump to any empty cell that is **exactly** *k* cells away from the current (to the left or to the right). Note that it doesn't matter whether intermediate cells are empty or not as the grasshopper makes a jump over them. For example, if *k* = 1 the grasshopper can jump to a neighboring cell only, and if *k* = 2 the grasshopper can jump over a single cell.

Your goal is to determine whether there is a sequence of jumps such that grasshopper will get from his initial position to the cell with an insect.

**Input**

The first line of the input contains two integers *n* and *k* (2 ≤ *n* ≤ 100, 1 ≤ *k* ≤ *n* - 1) — the number of cells in the line and the length of one grasshopper's jump.

The second line contains a string of length *n* consisting of characters '.', '#', 'G' and 'T'. Character '.' means that the corresponding cell is empty, character '#' means that the corresponding cell contains an obstacle and grasshopper can't jump there. Character 'G' means that the grasshopper starts at this position and, finally, 'T' means that the target insect is located at this cell. It's guaranteed that characters 'G' and 'T' appear in this line **exactly once**.

**Output**

If there exists a sequence of jumps (each jump of length *k*), such that the grasshopper can get from his initial position to the cell with the insect, print "YES" (without quotes) in the only line of the input. Otherwise, print "NO" (without quotes).

**Examples**

**input**

5 2  
#G#T#

**output**

YES

**input**

6 1  
T....G

**output**

YES

**input**

7 3  
T..#..G

**output**

NO

**input**

6 2  
..GT..

**output**

NO

**Note**

In the first sample, the grasshopper can make one jump to the right in order to get from cell 2 to cell 4.

In the second sample, the grasshopper is only able to jump to neighboring cells but the way to the insect is free — he can get there by jumping left 5 times.

In the third sample, the grasshopper can't make a single jump.

In the fourth sample, the grasshopper can only jump to the cells with odd indices, thus he won't be able to reach the insect.

#include <bits/stdc++.h>

using namespace std;

#define REP(i,a,b) for(int i=(a); i<=(b); ++i)

#define FOR(i,a,b) for(int i=(a); i>=(b); --i)

#define mp make\_pair

#define pb push\_back

#define ll long long

#define str string

const int N=1e3+100;

int n, s, t, k;

bool ok, fre[N];

char ch;

void home() {

#ifndef ONLINE\_JUDGE

freopen("382A.inp","r",stdin);

freopen("382A.out","w",stdout);

#endif

ios\_base::sync\_with\_stdio(false);

cin.tie(0);

cout.tie(0);

}

void Docdl() {

cin>>n>>k;

REP(i,1,n) {

cin>>ch;

if (ch=='G') s=i;

else

if (ch=='T') t=i;

if (ch=='#') fre[i]=false;

else fre[i]=true;

}

}

void Solve() {

if (abs(s-t)%k!=0) cout<<"NO";

else {

if (s>t) {

ok=true;

while (s!=t) {

s-=k;

if (!fre[s]) {

ok=false;

break;

}

}

if (ok) cout<<"YES";

else cout<<"NO";

}

else {

ok=true;

while (s!=t) {

s+=k;

if (!fre[s]) {

ok=false;

break;

}

}

if (ok) cout<<"YES";

else cout<<"NO";

}

}

}

int main() {

home();

Docdl();

Solve();

return 0;

}

<http://codeforces.com/contest/690/problem/D1>

D1. The Wall (easy)

time limit per test

0.5 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

*"The zombies are lurking outside. Waiting. Moaning. And when they come..."*

*"When they come?"*

*"I hope the Wall is high enough."*

Zombie attacks have hit the Wall, our line of defense in the North. Its protection is failing, and cracks are showing. In places, gaps have appeared, splitting the wall into multiple segments. We call on *you* for help. Go forth and explore the wall! Report how many disconnected segments there are.

The wall is a two-dimensional structure made of bricks. Each brick is one unit wide and one unit high. Bricks are stacked on top of each other to form columns that are up to *R* bricks high. Each brick is placed either on the ground or directly on top of another brick. Consecutive non-empty columns form a *wall segment*. The entire wall, all the segments and empty columns in-between, is *C* columns wide.

**Input**

The first line of the input consists of two space-separated integers *R* and *C*, 1 ≤ *R*, *C* ≤ 100. The next *R* lines provide a description of the columns as follows:

* each of the *R* lines contains a string of length *C*,
* the *c*-th character of line *r* is B if there is a brick in column *c* and row *R* - *r* + 1, and . otherwise.

The input will contain at least one character B and it will be valid.

**Output**

The number of wall segments in the input configuration.

**Examples**

**input**

3 7  
.......  
.......  
.BB.B..

**output**

2

**input**

4 5  
..B..  
..B..  
B.B.B  
BBB.B

**output**

2

**input**

4 6  
..B...  
B.B.BB  
BBB.BB  
BBBBBB

**output**

1

**input**

1 1  
B

**output**

1

**input**

10 7  
.......  
.......  
.......  
.......  
.......  
.......  
.......  
.......  
...B...  
B.BB.B.

**output**

3

**input**

8 8  
........  
........  
........  
........  
.B......  
.B.....B  
.B.....B  
.BB...BB

**output**

2

**Note**

In the first sample case, the 2nd and 3rd columns define the first wall segment, and the 5th column defines the second.

#include <bits/stdc++.h>

using namespace std;

typedef long long ll;

typedef pair<int,int> ii;

int main(int argc, char const \*argv[]) {

#ifndef ONLINE\_JUDGE

freopen("in", "r", stdin);

#endif

int n,m; cin>>n>>m; string str;

while(n--) cin>>str;

int i=0,cnt= str[0] == 'B';

for(int i=1; i<m; i++) {

if(str[i] == 'B' && str[i-1] == '.') cnt++;

} cout<<cnt<<endl;

}

<http://codeforces.com/contest/721/problem/A>

A. One-dimensional Japanese Crossword

time limit per test

1 second

memory limit per test

256 megabytes

input

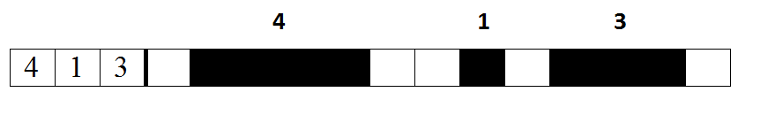
standard input

output

standard output

Recently Adaltik discovered japanese crosswords. Japanese crossword is a picture, represented as a table sized *a* × *b* squares, and each square is colored white or black. There are integers to the left of the rows and to the top of the columns, encrypting the corresponding row or column. The number of integers represents how many groups of black squares there are in corresponding row or column, and the integers themselves represents the number of consecutive black squares in corresponding group (you can find more detailed explanation in Wikipedia <https://en.wikipedia.org/wiki/Japanese_crossword>).

Adaltik decided that the general case of japanese crossword is too complicated and drew a row consisting of *n* squares (e.g. japanese crossword sized 1 × *n*), which he wants to encrypt in the same way as in japanese crossword.

The example of encrypting of a single row of japanese crossword.

Help Adaltik find the numbers encrypting the row he drew.

**Input**

The first line of the input contains a single integer *n* (1 ≤ *n* ≤ 100) — the length of the row. The second line of the input contains a single string consisting of *n* characters 'B' or 'W', ('B' corresponds to black square, 'W' — to white square in the row that Adaltik drew).

**Output**

The first line should contain a single integer *k* — the number of integers encrypting the row, e.g. the number of groups of black squares in the row.

The second line should contain *k* integers, encrypting the row, e.g. corresponding to sizes of groups of consecutive black squares in the order from left to right.

**Examples**

**input**

3  
BBW

**output**

1  
2

**input**

5  
BWBWB

**output**

3  
1 1 1

**input**

4  
WWWW

**output**

0

**input**

4  
BBBB

**output**

1  
4

**input**

13  
WBBBBWWBWBBBW

**output**

3  
4 1 3

**Note**

The last sample case correspond to the picture in the statement.

#include <bits/stdc++.h>

using namespace std;

#define IOS ios::sync\_with\_stdio(0); cin.tie(0); cout.tie(0);

#define endl "\n"

#define Max(x,y,z) max(x,max(y,z))

#define Min(x,y,z) min(x,min(y,z))

#define fr(i,s,e) for(i=s;i<e;i++)

#define rf(i,s,e) for(i=s-1;i>=e;i--)

#define pb push\_back

#define mp make\_pair

typedef long long ll;

typedef unsigned int uint;

typedef unsigned long long ull;

typedef pair<int, int> pii;

typedef pair<long long, long long> pll;

typedef vector<int> vi;

typedef vector<long long> vll;

typedef vector<pair<long long,long long> > vpll;

typedef vector<vector<ll> > vvll;

#define PI 3.141592653589793

#define MOD 1000000007

int main()

{

ll n;

cin>>n;

string s;

cin>>s;

ll count=0;

vector<ll> v;

ll len=0;

if(s[0]=='B')

{

count++;

len++;

}

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

{

if(s[i]=='B')

{

if(s[i-1]=='B')

{

len++;

}

else

{

len=1;

count++;

}

}

else

{

if(len>=1)

{

v.push\_back(len);

}

len=0;

}

}

if(len>=1)

{

v.push\_back(len);

}

cout<<count<<endl;

for(ll i=0;i<count;i++)

{

cout<<v[i]<<" ";

}

return 0;

}

<http://codeforces.com/contest/486/problem/A>

A. Calculating Function

time limit per test

1 second

memory limit per test

256 megabytes

input

standard input

output

standard output

For a positive integer *n* let's define a function *f*:

*f*(*n*) =  - 1 + 2 - 3 + .. + ( - 1)*nn*

Your task is to calculate *f*(*n*) for a given integer *n*.

**Input**

The single line contains the positive integer *n* (1 ≤ *n* ≤ 1015).

**Output**

Print *f*(*n*) in a single line.

**Examples**

**input**

4

**output**

2

**input**

5

**output**

-3

**Note**

*f*(4) =  - 1 + 2 - 3 + 4 = 2

*f*(5) =  - 1 + 2 - 3 + 4 - 5 =  - 3

#include<iostream>

#include<vector>

#include<algorithm>

using namespace std;

int main(){

long long n; cin >> n;

long long sum = 0;

long long num\_odd;

long long num\_even;

if(n % 2 == 0){

num\_even = n / 2;

num\_odd = num\_even;

}else{

num\_even = (n - 1)/2;

num\_odd = num\_even + 1;

}

cout << num\_even \* (num\_even + 1) - (num\_odd \* num\_odd);

}

<http://codeforces.com/contest/20/problem/C>

C. Dijkstra?

time limit per test

1 second

memory limit per test

64 megabytes

input

standard input

output

standard output

You are given a weighted undirected graph. The vertices are enumerated from 1 to *n*. Your task is to find the shortest path between the vertex 1 and the vertex *n*.

**Input**

The first line contains two integers *n* and *m* (2 ≤ *n* ≤ 105, 0 ≤ *m* ≤ 105), where *n* is the number of vertices and *m* is the number of edges. Following *m* lines contain one edge each in form *ai*, *bi* and *wi* (1 ≤ *ai*, *bi* ≤ *n*, 1 ≤ *wi* ≤ 106), where *ai*, *bi* are edge endpoints and *wi* is the length of the edge.

It is possible that the graph has loops and multiple edges between pair of vertices.

**Output**

Write the only integer -1 in case of no path. Write the shortest path in opposite case. If there are many solutions, print any of them.

**Examples**

**input**

5 6  
1 2 2  
2 5 5  
2 3 4  
1 4 1  
4 3 3  
3 5 1

**output**

1 4 3 5

**input**

5 6  
1 2 2  
2 5 5  
2 3 4  
1 4 1  
4 3 3  
3 5 1

**output**

1 4 3 5

#include <iostream>

#include <cstdio>

#include <set>

#include <vector>

#include <queue>

#include <climits>

#define MAX LONG\_LONG\_MAX

#define N 1000004

#define ll long long

using namespace std;

ll dist[N];

vector<pair<ll, int> > v[N];

priority\_queue<pair<ll, int> >adj;

int connected[N] = {};

//vector<int> x;

int prev[N] = {};

int ans[N] = {};

void find(int n){

while(connected[n] == 0){

int ind = adj.top().second;

int size = v[ind].size();

for(int i=0;i<size;i++){

if(connected[v[ind][i].second] == 0){

if(dist[v[ind][i].second] == MAX){

dist[v[ind][i].second] = dist[ind] + v[ind][i].first;

adj.push(make\_pair(-1 \* dist[v[ind][i].second], v[ind][i].second));

prev[v[ind][i].second] = ind;

}

else if(dist[v[ind][i].second] > dist[ind] + v[ind][i].first){

dist[v[ind][i].second] = dist[ind] + v[ind][i].first;

adj.push(make\_pair(-1 \* dist[v[ind][i].second], v[ind][i].second));

prev[v[ind][i].second] = ind;

}

}

}

connected[ind] = 1;

adj.pop();

//x.push\_back(ind);

}

}

bool dfs(int a, int n){

int s = v[a].size();

ans[a] = 1;

for(int i=0;i<s;i++){

if(ans[v[a][i].second] == 0){

dfs(v[a][i].second,n);

}

}

if(ans[n] == 1){

return true;

}

return false;

}

int main(){

int n, m, a, b;

ll w;

cin>>n>>m;

while(m--){

cin>>a>>b>>w;

v[a].push\_back(make\_pair(w,b));

v[b].push\_back(make\_pair(w,a));

}

//cout << dfs(1,n)<<"\n";

if(dfs(1,n) == true){

//cout << false << "\n";

v[0].push\_back(make\_pair(MAX, 1));

for(int i=2;i<=n;i++){

dist[i] = MAX;

adj.push(make\_pair(-1 \* dist[i], i));

}

dist[1] = 0;

adj.push(make\_pair(0, 1));

find(n);

int size;

int i = n;

vector<int> arr;

while(i != 0){

arr.push\_back(i);

i =prev[i];

}

size = arr.size();

for(int i=size-1;i>=0;i--){

cout << arr[i]<< " " ;

}

cout << "\n";

}

else cout << -1 << "\n";

return 0;

}

<http://codeforces.com/contest/712/problem/B>

B. Memory and Trident

time limit per test

2 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

Memory is performing a walk on the two-dimensional plane, starting at the origin. He is given a string *s* with his directions for motion:

* An 'L' indicates he should move one unit left.
* An 'R' indicates he should move one unit right.
* A 'U' indicates he should move one unit up.
* A 'D' indicates he should move one unit down.

But now Memory wants to end at the origin. To do this, he has a special trident. This trident can replace any character in *s* with any of 'L', 'R', 'U', or 'D'. However, because he doesn't want to wear out the trident, he wants to make the minimum number of edits possible. Please tell Memory what is the minimum number of changes he needs to make to produce a string that, when walked, will end at the origin, or if there is no such string.

**Input**

The first and only line contains the string *s* (1 ≤ |*s*| ≤ 100 000) — the instructions Memory is given.

**Output**

If there is a string satisfying the conditions, output a single integer — the minimum number of edits required. In case it's not possible to change the sequence in such a way that it will bring Memory to to the origin, output -1.

**Examples**

**input**

RRU

**output**

-1

**input**

UDUR

**output**

1

**input**

RUUR

**output**

2

**Note**

In the first sample test, Memory is told to walk right, then right, then up. It is easy to see that it is impossible to edit these instructions to form a valid walk.

In the second sample test, Memory is told to walk up, then down, then up, then right. One possible solution is to change *s* to "LDUR". This string uses 1 edit, which is the minimum possible. It also ends at the origin.

#include <bits/stdc++.h>

#define sqr(x) (x \* x)

#define cub(x) sqr(x) \* (x)

#define endl "\n"

#define all(x) x.begin() , x.end()

#define fp pop\_front

#define bp pop\_back

#define pf push\_front

#define pb push\_back

#define mp make\_pair

#define sz size()

#define se second

#define fi first

#define \_ std::ios\_base::sync\_with\_stdio(false); std::cin.tie(0); std::cout.tie(0);

#define rep(i, x, y) for (long long i = x; i <= y; ++i)

#define per(i, y, x) for (long long i = y; i >= x; --i)

using namespace std;

typedef long long ll;

typedef double dd;

typedef string st;

typedef char ch;

typedef unsigned long long ull;

typedef bool bl;

typedef long double ld;

typedef int I;

const ld pi = 3.14159265358979323846;

const ll mod = 1e9 + 7;

const ll N = 1e4;

const ll INF = 1e9 + 1;

ll ans, res;

st s;

int main () { \_

cin >> s;

if (s.sz % 2 != 0) {

cout << -1;

return 0;

}

rep (i, 0, s.sz - 1) {

if (s[i] == 'D') {

ans++;

}

if (s[i] == 'U') {

ans--;

}

if (s[i] == 'R') {

res++;

}

if (s[i] == 'L') {

res--;

}

}

cout << (abs(ans) + abs(res)) / 2;

return 0;

}