Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Журнал по исследовательской практике (индивидуальный план)

Команда: MAI #44: Vakhramyan, Kosogorov, Simonov

Студент	Группа
Косогоров Владислав Валерьевич	М8О-306Б-18
Вахрамян Кирилл Олегович	М8О-306Б-18
Симонов Сергей Яковлевич	М8О-306Б-18

Москва, 2021

Сводная таблица

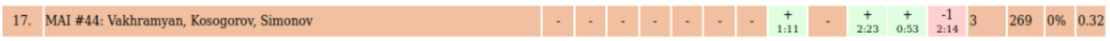
Дата	Название	Время	Место проведения	Решенные задачи
14.02.2021	XXI Open Cup named after E.V. Pankratiev. Grand Prix of Belarus, Division 2.	11:00-16:00	Дистанционно	N, P, Q
21.02.2021	XXI Open Cup named after E.V. Pankratiev. Grand prix of Suwon, Division 2.	11:00-16:00	Дистанционно	M, O, P
28.02.2021	XXI Open Cup named after E.V. Pankratiev. Grand prix of Tokyo, Division 2.	11:00-16:00	Дистанционно	K, L, N
14.03.2021	20-21 Своя тренировка 2: Semi-FFT	11:00-16:00	Дистанционно	G, I, J
21.03.2021	20-21 Своя тренировка 3: Graphs	11:00-16:00	Дистанционно	C, K
04.04.2021	2020-2021 ICPC, NERC, Northern Eurasia Onsite	11:00-16:00	Дистанционно	K
11.04.2021	20-21 Своя тренировка 4	11:00-16:00	Дистанционно	B, D, F
25.04.2021	RuCode Spring 2021 Championship C-E	11:00-16:00	Дистанционно	A, B
02.05.2021	XXI Open Cup named after E.V. Pankratiev. Grand prix of China, Division 2.	11:00-16:00	Дистанционно	O, P, Q, R
09.05.2021	XXI Open Cup named after E.V. Pankratiev. Grand Prix of Urals, Division 2	11:00-16:00	Дистанционно	A, N, O
23.05.2021	XXI Open Cup named after E.V. Pankratiev. Grand Prix of Southern Europe, Division 2	16:00-21:00	Дистанционно	Q

Явка

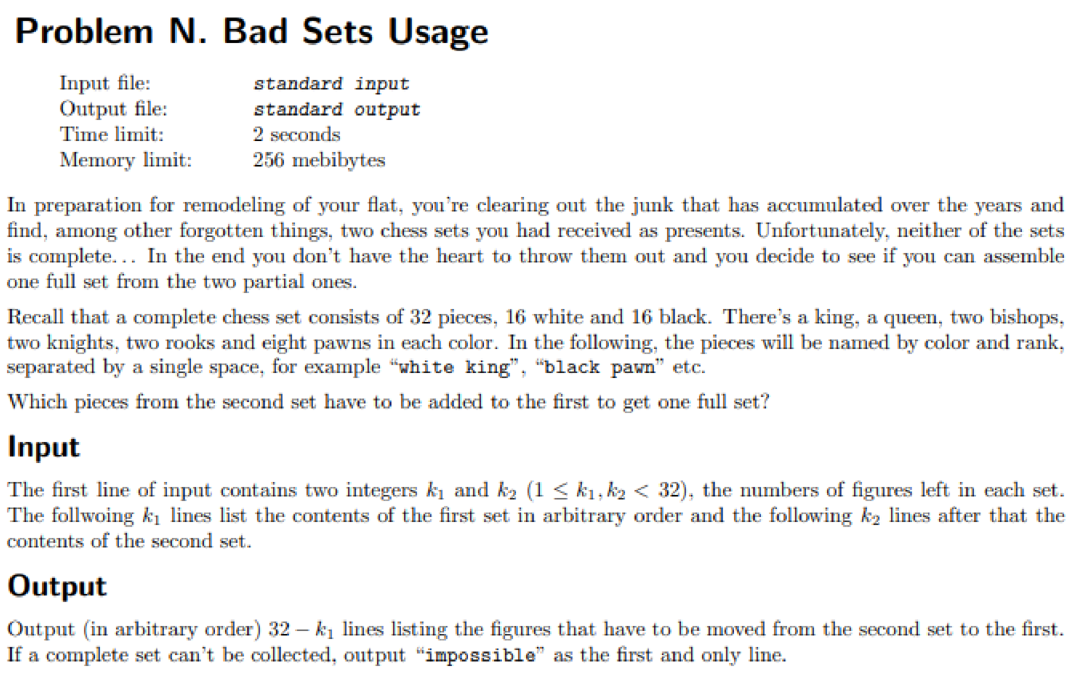
Дата	Название	Присутствующие
14.02.2021	XXI Open Cup named after E.V. Pankratiev. Grand Prix of Belarus, Division 2.	Вахрамян, Косогоров, Симонов
21.02.2021	XXI Open Cup named after E.V. Pankratiev. Grand Prix of Suwon, Division 2.	Вахрамян, Косогоров, Симонов
28.02.2021	XXI Open Cup named after E.V. Pankratiev. Grand prix of Tokyo, Division 2.	Вахрамян, Косогоров, Симонов
14.03.2021	20-21 Своя тренировка 2: Semi-FFT	Вахрамян, Косогоров, Симонов
21.03.2021	20-21 Своя тренировка 3: Graphs	Вахрамян, Косогоров, Симонов
04.04.2021	2020-2021 ICPC, NERC, Northern Eurasia Onsite	Вахрамян, Косогоров, Симонов
11.04.2021	20-21 Своя тренировка 4	Вахрамян, Косогоров, Симонов
25.04.2021	RuCode Spring 2021 Championship C-E	Вахрамян, Косогоров, Симонов
02.05.2021	XXI Open Cup named after E.V. Pankratiev. Grand prix of China, Division 2	Вахрамян, Косогоров, Симонов
09.05.2021	XXI Open Cup named after E.V. Pankratiev. Grand Prix of Urals, Division 2	Вахрамян, Косогоров, Симонов
23.05.2021	XXI Open Cup named after E.V. Pankratiev. Grand Prix of Southern Europe, Division 2	Вахрамян, Косогоров, Симонов

Условия и решения задач

XXI Open Cup named after E.V. Pankratiev. Grand Prix of Belarus



17.	MAI #44: Vakhramyan, Kosogorov, Simonov	-	-	-	-	-	-	-	+ 1:11	-	+ 2:23	+ 0:53	-1 2:14	3	269	0%	0.32



Problem N. Bad Sets Usage

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

In preparation for remodeling of your flat, you're clearing out the junk that has accumulated over the years and find, among other forgotten things, two chess sets you had received as presents. Unfortunately, neither of the sets is complete... In the end you don't have the heart to throw them out and you decide to see if you can assemble one full set from the two partial ones.

Recall that a complete chess set consists of 32 pieces, 16 white and 16 black. There's a king, a queen, two bishops, two knights, two rooks and eight pawns in each color. In the following, the pieces will be named by color and rank, separated by a single space, for example "`white king`", "`black pawn`" etc.

Which pieces from the second set have to be added to the first to get one full set?

Input

The first line of input contains two integers k_1 and k_2 ($1 \leq k_1, k_2 < 32$), the numbers of figures left in each set. The follwoing k_1 lines list the contents of the first set in arbitrary order and the following k_2 lines after that the contents of the second set.

Output

Output (in arbitrary order) $32 - k_1$ lines listing the figures that have to be moved from the second set to the first. If a complete set can't be collected, output "`impossible`" as the first and only line.

Решение

```
k1, k2 = [int(i) for i in input().split()]

w_1 = [0, 0, 0, 0, 0, 0]
b_1 = [0, 0, 0, 0, 0, 0]
w_2 = [0, 0, 0, 0, 0, 0]
b_2 = [0, 0, 0, 0, 0, 0]

res = []
text = list()

for i in range(k1):
        text = input().split(' ')
        if (text[0] == "black"):
                if (text[1] == "king"):
                        b_1[0] += 1
                if (text[1] == "queen"):
                        b_1[1] += 1
                if (text[1] == "bishop"):
                        b_1[2] += 1
                if (text[1] == "knight"):
                        b_1[3] += 1
                if (text[1] == "rook"):
                        b_1[4] += 1
                if (text[1] == "pawn"):
                        b_1[5] += 1
        if (text[0] == "white"):
                if (text[1] == "king"):
                        w_1[0] += 1
                if (text[1] == "queen"):
                        w_1[1] += 1
                if (text[1] == "bishop"):
                        w_1[2] += 1
                if (text[1] == "knight"):
                        w_1[3] += 1
                if (text[1] == "rook"):
                        w_1[4] += 1
                if (text[1] == "pawn"):
                        w_1[5] += 1

for i in range(k2):
        text = input().split(' ')
        if (text[0] == "black"):
                if (text[1] == "king"):
                        b_2[0] += 1
                if (text[1] == "queen"):
                        b_2[1] += 1
                if (text[1] == "bishop"):
                        b_2[2] += 1
                if (text[1] == "knight"):
                        b_2[3] += 1
                if (text[1] == "rook"):
                        b_2[4] += 1
                if (text[1] == "pawn"):
                        b_2[5] += 1
        if (text[0] == "white"):
                if (text[1] == "king"):
                        w_2[0] += 1
                if (text[1] == "queen"):
                        w_2[1] += 1
        if (text[1] == "bishop"):
            w_2[2] += 1
        if (text[1] == "knight"):
            w_2[3] += 1
        if (text[1] == "rook"):
            w_2[4] += 1
        if (text[1] == "pawn"):
            w_2[5] += 1

impossible_situation = 0

if (impossible_situation == 0 and w_1[0] < 1):
    if (w_2[0] < 1):
        print("impossible")
        impossible_situation = 1
    else:
        res.append("white king")

if (impossible_situation == 0 and w_1[1] < 1):
    if (w_2[1] < 1):
        print("impossible")
        impossible_situation = 1
    else:
        res.append("white queen")

if (impossible_situation == 0 and w_1[2] < 2):
    if (w_2[2] + w_1[2] < 2):
        print("impossible")
        impossible_situation = 1
    else:
        for i in range(2 - w_1[2]):
            res.append("white bishop")

if (impossible_situation == 0 and w_1[3] < 2):
    if (w_2[3] + w_1[3] < 2):
        print("impossible")
        impossible_situation = 1
    else:
        for i in range(2 - w_1[3]):
            res.append("white knight")

if (impossible_situation == 0 and w_1[4] < 2):
    if (w_2[4] + w_1[4] < 2):
        print("impossible")
```

```
                impossible_situation = 1
        else:
                for i in range(2 - w_1[4]):
                        res.append("white rook")

if (impossible_situation == 0 and w_1[5] < 8):
        if (w_2[5] + w_1[5] < 8):
                print("impossible")
                impossible_situation = 1
        else:
                for i in range(8 - w_1[5]):
                        res.append("white pawn")

if (impossible_situation == 0 and b_1[0] < 1):
        if (b_2[0] < 1):
                print("impossible")
                impossible_situation = 1
        else:
                res.append("black king")

if (impossible_situation == 0 and b_1[1] < 1):
        if (b_2[1] < 1):
                print("impossible")
                impossible_situation = 1
        else:
                res.append("black queen")

if (impossible_situation == 0 and b_1[2] < 2):
        if (b_2[2] + b_1[2] < 2):
                print("impossible")
                impossible_situation = 1
        else:
                for i in range(2 - b_1[2]):
                        res.append("black bishop")

if (impossible_situation == 0 and b_1[3] < 2):
        if (b_2[3] + b_1[3] < 2):
                print("impossible")
                impossible_situation = 1
        else:
                for i in range(2 - b_1[3]):
                        res.append("black knight")

if (impossible_situation == 0 and b_1[4] < 2):
        if (b_2[4] + b_1[4] < 2):
                print("impossible")
                impossible_situation = 1
        else:
                for i in range(2 - b_1[4]):
                        res.append("black rook")

if (impossible_situation == 0 and b_1[5] < 8):
        if (b_2[5] + b_1[5] < 8):
                print("impossible")
                impossible_situation = 1
        else:
                for i in range(8 - b_1[5]):
                        res.append("black pawn")

if (impossible_situation == 0):
        for i in res:
                print(i)
else:
    pass
```

Problem P. Bored Serving Users?

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

The early 2000's...

Having just boasted about your extensive experience in designing small-scale home networks, you were asked to take on the job for your apartment block. Now is probably not the time to confess that this is really your first attempt.

So, the network will connect N users. Each of them connects to one of the hubs via a dedicated network cable. The hubs may also be connected to one another with similar cables. Any socket of a hub may be used by either an end user or for a connection to another hub. The final network will have to enable any pair of users to communicate with each other via one or more hubs.

Digging through your piles of old hardware, you found M hubs suitable for the purpose. The i^{th} of those hubs $(1 \le i \le M)$ has k_i sockets for network cables.

As you have promised that each end user will only have pay for the cable to connect themselves to the nearest hub, you want to design the network using minimal number of your hubs (so you can keep the rest for future projects).

Are you up to the task?

Input

The first input line contains the integers N and M $(2 \le N \le 1000, 1 \le M \le 300)$. The second line contains M integers, the values of k_i $(2 \le k_i \le 48)$.

Output

If there is no solution, the only line of output should contain the text `Epic fail`.

Otherwise, the first line of output should contain K, the number of hubs used, and the second line K integers, the numbers of the hubs (they are numbered starting from one in the order they are given in the input).

If there are several solutions, output any one of them.

Решение

```
#include <bits/stdc++.h>

using namespace std;


bool cmp(pair<int, int> & a, pair<int, int> & b) {
    return a.second > b.second;
}


int main() {
    int N, M;
    cin >> N >> M;
    vector<int> ans;

    vector<pair<int, int>> k(M);
    for (int i = 0; i < M; i++) {
        pair<int, int> tmp;
        tmp.first = i + 1;
        cin >> tmp.second;
        k[i] = tmp;
    }
    sort(k.begin(), k.end(), cmp);
    if (k[0].second >= N) {
        cout << 1 << '\n' << k[0].first << '\n';
    } else {
        int sum = k[0].second;
        ans.push_back(k[0].first);
        int i = 1;
        while (1 and i < M) {
            sum += k[i].second - 2;
            ans.push_back(k[i].first);
            i++;
            if (sum >= N) {
                break;
            }
        }
        if (sum >= N) {
            cout << ans.size() << '\n';
            for (auto a : ans) {
                cout << a << ' ';
            }
            cout << '\n';
        }
        else {
            cout << "Epic fail\n";
}
    }

}
```

Problem Q. Base Structure Under...

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

A well-known software development company has been commissioned by the Archaeological Society. One of the modules has to help the archaeologists to process data about the ruins of buildings they have found during their excavations of ancient cities. Development of this module has been assigned to Vasya.

Vasya, being a seasoned programmer, at once noticed that the module would need a database to contain the descriptions of the ruins and the estimated construction times of the buildings. It would be all fine, but suddenly the manager got the genial idea that since the database describes Roman ruins, the years of construction should be stored in the Roman number system. Now Vasya is wondering how many symbols he needs to set aside for each year number field in the database. According to the functional specification, the software module must be able to handle years from A to B (inclusive). Help Vasya determine the minimal number of characters sufficient for storing any year number in the range from A to B.

Input

The only input line contains the descriptions of the years A and B, separated by the "-" sign. A description of a year consists of one to four decimal digits (the number of the year), followed by either "AD" (*Anno Domini*, the current era) or "BC" (*Before Christ*, before the current era). In both directions the years are numbered starting from 1. It is known that 753BC $\le A \le B \le$ 2012AD.

Output

The output should consist of a single integer, the minimal number of characters that have to be reserved in the database for the year number.

Решение

```
def count(n):
        if (n == 0):
                return 0
        elif (n == 1):
                return 1
        elif (n == 2):
                return 2
        elif (n == 3):
                return 3
        elif (n == 4):
                return 2
        elif (n == 5):
                return 1
        elif (n == 6):
                return 2
        elif (n == 7):
                return 3
        elif (n == 8):
                return 4
        elif (n == 9):
                return 2
```

```
year_1, year_2 = input().split('-')

if (year_1[len(year_1)-2 : len(year_1)] == 'BC'):
        decimal_1 = 754 - int(year_1[0 : len(year_1)-2])
elif (year_1[len(year_1)-2 : len(year_1)] == 'AD'):
        decimal_1 = int(year_1[0 : len(year_1)-2]) + 753

if (year_2[len(year_2)-2 : len(year_2)] == 'BC'):
        decimal_2 = 754 - int(year_2[0 : len(year_2)-2])
elif (year_2[len(year_2)-2 : len(year_2)] == 'AD'):
    decimal_2 = int(year_2[0 : len(year_2)-2]) + 753

res = 0

for i in range(decimal_1, decimal_2 + 1):
        j = i
        size1 = 0
        while j > 0:
                size1 += count(j % 10)
                j = int(j / 10)
        if (size1 > res):
                res = size1

print(res)
```

17.	MAI #44: Vakhramyan, Kosogorov, Simonov	-	-	-	-	-	-	-	+ 1:11	-	+ 2:23	+ 0:53	-1 2:14	3	269	0%	0.32

Problem M. Move To The Equality

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	1024 mebibytes

Take any four positive integers: a, b, c, d. Form four more, like this: $|a-b|$ $|b-c|$ $|c-d|$ $|d-a|$. Then, do it again with these four new numbers. And then again. And again. Eventually, all four integers will be the same.

Given a, b, c and d, figure out how quickly the sequence converges.

Input

There will be no more than 1100 several test cases in the input.

Each test case consists of four positive integers on a single line ($1 \le a, b, c, d \le 2 \cdot 10^9$) in that order. The input will end with a line with four zeros, which should not be processed.

Output

For each test case, print a single integer on its own line, indicating the number of steps until convergence.

Решение

```
#include <iostream>

using namespace std;

long long my_abs(long long a, long long b) {
    long long res = a - b;
    if (res >= 0) {
        return res;
    }
    return -res;
}

int main() {
    long long a, b, c, d;
    cin >> a >> b >> c >> d;

    while (a && b && c && d) {
        long long conv_steps = 0;

        while (true) {
            if (a == b && b == c && c == d) {
                break;
            }
            long long a_1 = a;
            long long b_1 = b;
            long long c_1 = c;
            long long d_1 = d;
            a = my_abs(a_1, b_1);
            b = my_abs(b_1, c_1);
            c = my_abs(c_1, d_1);
            d = my_abs(d_1, a_1);
            // cout << a << ' '<< b << ' ' << c << ' ' << d << '\n';
            conv_steps += 1;
        }

        cout << conv_steps << endl;
        cin >> a >> b >> c >> d;
    }

}
```

Problem O. Olympic Tournament

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	1024 mebibytes

Alex missed the figure skating olympic qualification that he wanted to attend. So now he wants to know the pairs of skaters whose dancing he missed. He had several photos from the warmup, so he chose one where all skaters are clearly visible and wrote down the coordinates of all N skaters (N is even).

Then Alex determined the pairs of skaters by the following algorithm: from not yet paired skaters he chooses two closest (to each other) skaters and assumes that they dance together as a pair. Should he find several pairs of skaters with the same minimum distance between skaters, he chooses lexicographically smallest pair (Alex enumerated skaters by integers from 1 to N, skaters are ordered inside a pair, one with lower number goes first). You are asked to help Alex to determine pairs of skaters.

Input

The first line of input contains even integer N ($2 \le N \le 300$). Each i-th line of the next N lines contains two integers — x and y coordinates of point, representing i-th skater. All points are distinct. All coordinates are less than 10^8 by absolute value.

Output

You should output $N/2$ lines. Each line must contain numbers of skaters in the corresponding pair. The first number in a line should be less than the second. Lines must be sorted in the lexicographically ascending order.

Решение

```
#include <iostream>
#include <map>
```

```
#include <cmath>
#include <algorithm>
#include <limits>
#include <vector>

using namespace std;

int main() {
    int N;
    cin >> N;

    map<int, pair<double, double> > skater;
    for (int i = 1; i <= N; ++i) {
        double x, y;
        cin >> x >> y;
        skater[i] = {x, y};
    }

    map<pair<int, int>, double > distances;
    for (int i = 1; i < N; ++i) {
        double dist;

        for (int j = i + 1; j <= N; ++j) {
            distances[{i, j}] = sqrt(pow(skater[i].first - skater[j].first, 2)
            + pow(skater[i].second - skater[j].second, 2));
        }

    }

    vector<pair<int, int> > ans;
    while (!distances.empty()) {
        double min_dist = numeric_limits<double>::max();
        for (auto elem : distances) {
            if (elem.second < min_dist) {
                min_dist = elem.second;
                // cout << min_dist << endl;
            }
        }

        vector<pair<int, int> > min_pairs;
        for (auto it = distances.begin(); it != distances.end(); ++it) {
            if (it->second == min_dist) {
                int a = it->first.first;
                int b = it->first.second;
                if (a <= b) {
                    min_pairs.push_back({a, b});
                } else {
                    min_pairs.push_back({b, a});
                }
            }
        }

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

        ans.push_back(min_pairs[0]);
        // cout << min_pairs[0].first << ' ' << min_pairs[0].second << endl;

        vector<pair<int, int> > to_erase;
        for (auto elem : distances) {
            if (elem.first.first == min_pairs[0].first || elem.first.first ==
                min_pairs[0].second ||
                        elem.first.second == min_pairs[0].first ||
                        elem.first.second == min_pairs[0].second) {
                    to_erase.push_back({elem.first.first, elem.first.second});
                        }
        }
        min_pairs.clear();

        for (auto elem : to_erase) {
            distances.erase(elem);
        }

        to_erase.clear();
    }

    sort(ans.begin(), ans.end());
    for (auto elem : ans) {
        cout << elem.first << ' ' << elem.second << endl;
    }
}
```

Problem P. Product Game

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	1024 mebibytes

Rules of The Product game are simple. Master announces positive integer N. Player needs to calculate product of digits for each positive integer up to N and report the greatest product.

All answers should be known beforehand to allow interactive communication during a game. The Game Master asks you to write a program which having positive integer N will find correct answer.

Input

Input file contains one integer N ($1 \le N \le 2 \cdot 10^9$).

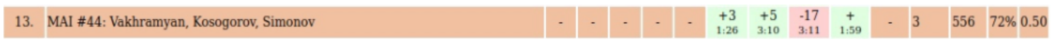
Output

Output file should contain greatest product for given N.

Решение

```
#include <bits/stdc++.h>


int maxProduct(int A) {
    if (A == 0)
        return 1;
    if (A < 10)
        return A;
    return std::max(maxProduct(A / 10) * (A % 10), maxProduct(A / 10 - 1) * 9);
}

int main() {
    size_t A;
    std::cin >> A;
    std::cout << maxProduct(A) << std::endl;

    return 0;
}
```

XXI Open Cup named after E.V. Pankratiev. Grand Prix of Tokyo



13.	MAI #44: Vakhramyan, Kosogorov, Simonov	-	-	-	-	-	+3 1:26	+5 3:10	-17 3:11	+ 1:59	-	3	556	72%	0.50

Problem K. King And Highways

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	1024 megabytes

There are x cities in the kingdom. King plans to connect some cities with bidirectional highways (each highway connects exactly two cities) such as any there will be route between any two cities on the highways when no more than one highway is closed.

Print minimum number of roads to build 998244353.

Input

The first line contains an integer x $(3 \le x < 10^{(10^6)})$.

Output

Print the answer.

Решение

```
#include<bits/stdc++.h>
using ll=long long int;
using namespace std;

struct Bigint {
    string a;
    int sign;

    Bigint(){}
    void operator = (string b) {
        a= (b[0]=='-' ? b.substr(1) : b);
        reverse(a.begin(), a.end());
        (*this).Remove0(b[0]=='-' ? -1 : 1);
    }
    Bigint(string x) {(*this)=x;}
    Bigint(ll x) {(*this)=to_string(x);}
    void operator = (ll x){*this=to_string(x);}

    char operator[](int i){return a[i];}
    int size() {return a.size();}
    Bigint inverseSign() {sign*=-1; return (*this);}

    Bigint Remove0(int newSign) {
```

```
    sign = newSign;
    for(int i=a.size()-1; i>0 && a[i]=='0'; i--) a.pop_back();
    if(a.size()==1 && a[0]=='0') sign=1;
    return (*this);
}

bool operator == (Bigint x) {return sign==x.sign && a==x.a;}
bool operator == (string x) {return *this==Bigint(x);}
bool operator == (ll x)     {return *this==Bigint(x);}
bool operator != (Bigint x) {return !(*this==x);}
bool operator != (string x) {return !(*this==x);}
bool operator != (ll x)     {return !(*this==x);}

bool operator < (Bigint b) {
    if (sign!=b.sign) return sign<b.sign;
    if(a.size()!=b.size()) return a.size()*sign<b.size()*sign;
    for(int i=a.size()-1; i>=0; i--)
        if(a[i] != b[i]) return a[i]<b[i];
    return false;
}
bool operator <  (string x) {return *this<Bigint(x);}
bool operator <  (ll x)     {return *this<Bigint(x);}
bool operator <= (Bigint b) {return *this==b || *this<b;}
bool operator <= (string b) {return *this==b || *this<b;}
bool operator <= (ll b)     {return *this==b || *this<b;}
bool operator >  (Bigint b) {return !(*this==b || *this<b);}
bool operator >  (string x) {return !(*this==x || *this<x);}
bool operator >  (ll b)     {return !(*this==b || *this<b);}
bool operator >= (Bigint b) {return *this==b || *this>b;}
bool operator >= (string b) {return *this==b || *this>b;}
bool operator >= (ll b)     {return *this==b || *this>b;}

Bigint operator + (Bigint b) {
    if(sign != b.sign) return (*this)-b.inverseSign();
    Bigint sum;
    for(int i=0, carry=0; i<a.size() || i<b.size() || carry; i++){
        if (i<a.size()) carry+=a[i]-'0';
        if (i<b.size()) carry+=b[i]-'0';
        sum.a += (carry % 10 + 48);
        carry /= 10;
    }
    return sum.Remove0(sign);
}
Bigint operator +  (string x) {return *this+Bigint(x);}
Bigint operator +  (ll x)     {return *this+Bigint(x);}
Bigint operator ++ (int) {*this+=1; return *this-1;}
Bigint operator ++ ()    {*this+=1; return *this;}
  void operator += (Bigint x) {*this = *this+x;}
  void operator += (string x) {*this = *this+x;}
  void operator += (ll x)     {*this = *this+x;}


Bigint operator - ( Bigint b ) {
    if(sign != b.sign) return (*this)+b.inverseSign();
    if(*this < b) return (b - *this).inverseSign();
    Bigint sub;
    for(int i=0,borrow=0; i<a.size(); i++) {
        borrow = a[i]-borrow-(i<b.size() ? b.a[i] : '0');
        sub.a += borrow>=0 ? borrow+'0' : borrow + 58;
        borrow = borrow>=0 ? 0:1;
    }
    return sub.Remove0(sign);
}
Bigint operator - (string x) {return *this-Bigint(x);}
Bigint operator - (ll x)     {return *this-Bigint(x);}
Bigint operator -- (int) {*this-=1; return *this+1;}
Bigint operator -- ()    {*this-=1; return *this;}
  void operator -= (Bigint x) {*this = *this-x;}
  void operator -= (string x) {*this = *this-x;}
  void operator -= (ll x)     {*this = *this-x;}

Bigint operator * (Bigint b) {
    Bigint mult("0");
    for(int i=0, k=a[i]; i<a.size(); i++, k=a[i]) {
        while(k-- -'0') mult=mult+b;
        b.a.insert(b.a.begin(),'0');
    }
    return mult.Remove0(sign * b.sign);
}
Bigint operator * (string x) {return *this*Bigint(x);}
Bigint operator * (ll x)     {return *this*Bigint(x);}
  void operator *= (Bigint x) {*this = *this*x;}
  void operator *= (string x) {*this = *this*x;}
  void operator *= (ll x)     {*this = *this*x;}

Bigint operator / (Bigint b) {
    if(b.size()==1 && b[0]=='0') b.a[0]/=(b[0]-'0');
    Bigint c("0"), d;
    for(int j=0; j<a.size(); j++) d.a += "0";
    int dSign = sign*b.sign; b.sign=1;
    for(int i=a.size()-1; i>=0; i--) {
        c.a.insert(c.a.begin(),'0');
        c=c+a.substr(i,1);
        while(!(c<b)) c=c-b, d.a[i]++;
    }
    return d.Remove0(dSign);
}
Bigint operator / (string x) {return *this/Bigint(x);}
Bigint operator / (ll x)     {return *this/Bigint(x);}
  void operator /= (Bigint x) {*this = *this/x;}
  void operator /= (string x) {*this = *this/x;}
  void operator /= (ll x)     {*this = *this/x;}

Bigint operator % (Bigint b) {
    if( b.size()==1 && b[0]=='0') b.a[0]/=(b[0]-'0') ;
    Bigint c("0");
    int cSign = sign*b.sign; b.sign=1;
    for( int i=a.size()-1; i>=0; i-- ) {
        c.a.insert( c.a.begin(),'0');
        c = c+a.substr(i,1);
        while(!(c<b)) c=c-b;
    }
    return c.Remove0(cSign);
}
Bigint operator % (string x) {return *this%Bigint(x);}
Bigint operator % (ll x)     {return *this%Bigint(x);}
  void operator %= (Bigint x) {*this = *this%x;}
  void operator %= (string x) {*this = *this%x;}
  void operator %= (ll x)     {*this = *this%x;}

void print() {
    if(sign==-1) putchar('-');
    for(int i=a.size()-1; i>=0; i--) putchar(a[i]);
}
friend istream& operator >>(istream &in,Bigint &x){
    string s; in>>s; x=s; return in;
}
friend ostream& operator <<(ostream &out,Bigint &x){
    if(x.sign==-1) putchar('-');
    for(int i=x.size()-1; i>=0; i--)
        putchar(x[i]);
    return out;
    }

    friend Bigint pow(Bigint base, Bigint pw){
        Bigint ans=1;
        while(pw!=0){
            if(pw%2 !=0) ans*=base;
            base*=base, pw/=2;
        }
        return ans;
    }
    friend Bigint pow(Bigint a, Bigint b,Bigint mod) {
        if (b==0) return Bigint(1);
        Bigint tmp=pow(a,b/2,mod);
        if ((b%2)==0) return (tmp*tmp)%mod;
        else return (((tmp*tmp)%mod)*a)%mod;
    }
    friend Bigint sqrt(Bigint x) {
        Bigint ans=x,tmp=(x+1)/2;
        while (tmp<ans) ans=tmp, tmp=(tmp+x/tmp)/2;
        return ans;
    }
    friend Bigint gcd(Bigint a,Bigint b){
        return a%b==0 ? b : gcd(b, a%b);
    }
    friend Bigint lcm(Bigint a,Bigint b){
        return a/gcd(a,b);
    }
};

int main(){
    Bigint x, ans;
    cin >> x;
    ans = x % "998244353";
    cout << ans << endl;
```

Problem L. Looking For Plagiarism

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	1024 megabytes

A program for the Typhoon programming language is interpreted with the virtual machine.

The SP register of that machine is set to 0 before execution; each operator is interpreted either as the `write` type instruction (increases SP by 1) or as `read` type instruction (decreases SP by 1). At the end of execution value of SP shall be equal to zero (otherwise Run-Time Error is sent); when SP becomes negative, the program goes into the infinite loop (and Time-Limit Exceed is sent).

The students were asked to write a program on Typhoon, consisting of exactly n operators, where n is even. There is built-in antiplagiarism tool in the contest management system which checks the *operational profile* of the accepted solution (i.e when neither Run-Time Error nor Time-Limit Exceed were sent); the profile is defined as the sequence of the n values of SP register after execution of each command.

If those profiles coincide for two students, both solutions are rejected as plagiarism.

Given integer n, calculate the maximum number of accepted solutions in the system modulo 998244353.

Input

The first line contains an integer n ($2 \le n \le 640\,000$, n is **even**)

Output

Print the answer.

Решение

```
#include <bits/stdc++.h>
using ll = long long;
using namespace std;

ll MOD = 998244353;

ll binpow(ll a, ll n) {
    ll res = 1;
    while (n) {
        if (n & 1) {
            res *= a;
            res %= MOD;
        }
        a *= a;
        a %= MOD;
        n >>= 1;
    }
    return res;
}

ll reversed(ll n) {
    return binpow(n, MOD - 2);
}

ll catalan(ll n) {
    ll res = 1;
    ll i = 0;
    while (i < n) {
        ++i;
        res *= 2 * (2*i-1);
        res %= MOD;
        res *= reversed(i+1);
        res %= MOD;
    }
    return res;
}

int main() {
    ll n;
    cin >> n;
    cout << catalan(n/2) << endl;
}
```

Problem N. N-dimensional Game

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	1024 megabytes

Alice and Bob are playing some adventure game in the n-dimensional Euclidean space. They have full set of dices: all distinct convex regular n-dimensional polytopes with unit length of the edge. $n-1$-dimensional faces of each of those dices are enumerated by sequential integers, starting from 1. At one turn, player throws all dices and counts the sum on the faces they are staying on.

Given n, find maximum possible sum player can obtain with the maximal luck, modulo 998244353.

Input

The first line contains an integer n ($3 \le n \le 998244352$).

Output

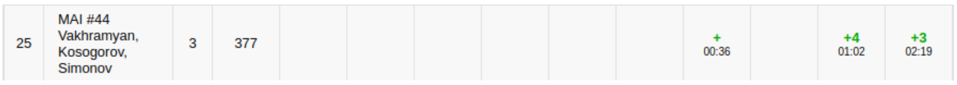
Print the answer.

Решение

```
#include <iostream>

using namespace std;
using ll = long long;
```

```
long long my_pow(long long base, long long p) {
    if (p == 1) {
        return base;
    } else if (p % 2 == 0) {
        ll t = my_pow(base, p / 2);
        return t * t % 998244353;
    } else {
        return my_pow(base, p - 1) * base % 998244353;
    }
}

int main() {
    ll n, a = 0, b = 0, c = 0, d = 0;
    cin >> n;
    if (n == 3){
        cout << 60;
    } else if (n == 4){
        cout << 773;
    } else {
        a = (n + 1) % 998244353;
        b = (n * 2) % 998244353;
        d = (a + b) % 998244353;
        c = my_pow(2, n);
        cout << (d + c) % 998244353;
    }
}
```

20-21 Своя тренировка 2: Semi-FF

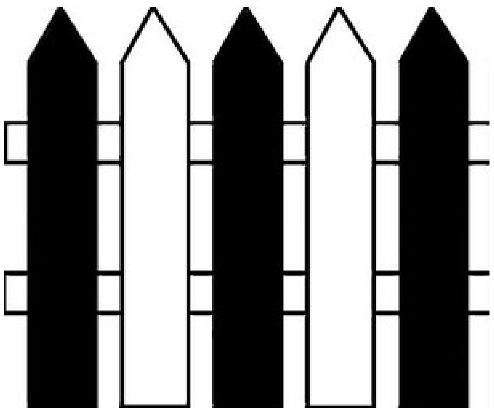


25	MAI #44 Vakhramyan, Kosogorov, Simonov	3	377							+ 00:36		+4 01:02	+3 02:19

G. Wooden Fence

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Asem has a wooden fence consisting of n boards, in which n is an odd number. Asem wants to paint this fence using two colors; black and white, such that the first board will be painted black, the second board will be painted white, the third board will be painted black, and so on.



Asem has black paint that can paint at most x boards and white paint that can paint at most y boards. Your task is to determine if you can paint the whole fence or not. Can you?

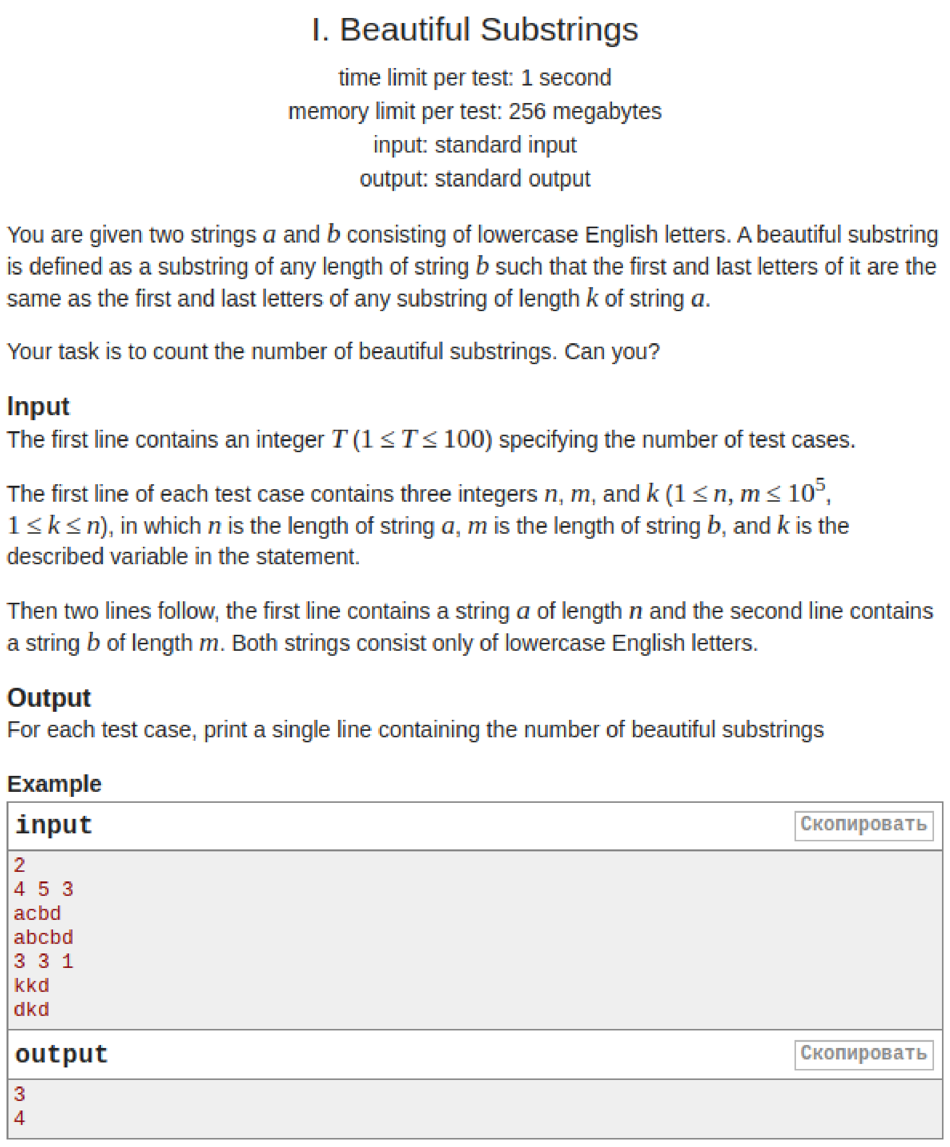
Input
The first line contains an integer T ($1 \le T \le 10^4$) specifying the number of test cases.

Each test case consists of a line containing three integers n, x, y ($1 \le n, x, y < 10^9$), in which n is an odd number specifying the number of boards in the fence, x is the maximum number of boards that can be painted in black, and y is the maximum number of boards that can be painted in white.

Output
For each test case, print a single line containing "YES" (without quotes) if you can paint the whole fence. Otherwise, print "NO" (without quotes).

Решение

```
#include <bits/stdc++.h>

using namespace std;

int main() {
    int t;
    cin >> t;
    while(t--) {
        uint64_t n, x, y;
        cin >> n >> x >> y;
        if (x > int(n / 2) and y >= int(n / 2))
            cout << "YES\n";
        else
            cout << "NO\n";
    }
}
```



I. Beautiful Substrings

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given two strings a and b consisting of lowercase English letters. A beautiful substring is defined as a substring of any length of string b such that the first and last letters of it are the same as the first and last letters of any substring of length k of string a.

Your task is to count the number of beautiful substrings. Can you?

Input
The first line contains an integer T ($1 \le T \le 100$) specifying the number of test cases.

The first line of each test case contains three integers n, m, and k ($1 \le n, m \le 10^5$, $1 \le k \le n$), in which n is the length of string a, m is the length of string b, and k is the described variable in the statement.

Then two lines follow, the first line contains a string a of length n and the second line contains a string b of length m. Both strings consist only of lowercase English letters.

Output
For each test case, print a single line containing the number of beautiful substrings

Example

input Скопировать

```
2
4 5 3
acbd
abcbd
3 3 1
kkd
dkd
```

output Скопировать

```
3
4
```

Решение

```
#include <cstdio>
#include <cstring>
#include <set>

using namespace std;
char a[100000], b[100000];

int main() {
    int T;
    scanf("%d", &T);
    while (T--) {
        int n, m, k;
        long long num = 0;
        scanf("%d %d %d", &n, &m, &k);
        scanf(" %s %s", a, b);

        set<pair<char, char> > s;
        for (int i = 0; i + k - 1 < n; i++) {
            s.insert(make_pair(a[i], a[i + k - 1]));
        }

        for (auto i : s) {
            char s = i.first, e = i.second;
            long long e_val = 0;
            if (s != e) {
                for (int j = m - 1; j >= 0; j--) {
                    if (b[j] == e) {
                        e_val++;
                    } else if (b[j] == s) {
                        num += e_val;
                    }
                }
            } else {
                for (int j = m - 1; j >= 0; j--) {
                    if (b[j] == e) {
                        e_val++;
                        num += (e_val);
                    }
                }
            }
        }
        printf("%lld\n", num);
    }
    return 0;
}
```

J. Stupid Submissions

time limit per test: 1.5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Abed is so motivated this year, his goal is qualifying for the 2019 ACM International Collegiate Programming Contest (ACM ICPC). Therefore, he always trains using Codeforces online judge. If you do not know Codeforces, the following rules can help you to understand how it works:

- Each problem has a set of n tests numbered from 1 to n. The first k tests are called sample tests. These tests are visible in the problem statement and a user can see them all.
- When Abed gets a wrong answer on test x ($x \le n$), he can see all the tests up to test number x. If Abed gets accepted, he can see all the tests.
- Each test can be either small or big. Abed can see the entire test only if it is small. If the test is big, Abed can see it partially. All the sample tests are small.

For example, let us consider a problem contains 6 tests, in which the first two tests are the sample tests. If Abed got a wrong answer on test 4, a wrong answer on test 3, and a wrong answer on test 5, he can see all tests from 1 to 5. If test 3 is small and the tests from 4 to 6 are big, Abed can see the first three tests fully, but he cannot fully see the remaining tests.

Unfortunately, Abed usually gets a lot of wrong answers. A submission made by Abed is called stupid if he got a wrong answer on a small test that he can see it fully. You are given a list of submissions made be Abed, your task is to count how many stupid submission Abed has made. Can you?

Input

The first line contains an integer T ($1 \le T \le 100$) specifying the number of test cases.

The first line of each test case contains three integers n, m, and k ($1 \le k \le n \le 10^4$, $1 \le m \le 10^4$), in which n is the number of tests in the problem, m is the number of submissions Abed has made, and k is the number of sample tests. Then a line follow contains n characters $t_1, \dots, t_n$, in which t_i is 'S' if the i^{th} test is small, and 'B' if it is big.

Then m lines follow, giving the list of submissions Abed has made, such that the i^{th} line will either contain a single character 'A' if Abed got accepted on the i^{th} submission, or contain a character 'W' and an integer x ($1 \le x \le n$) giving that Abed got a wrong answer on test case x.

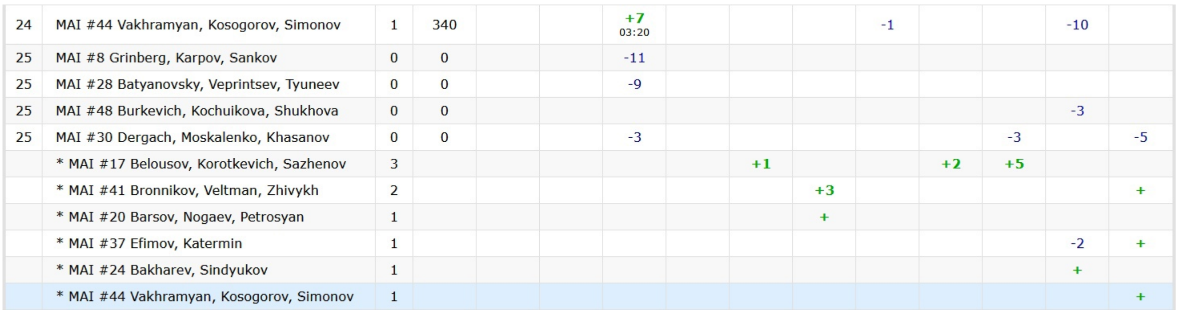
Output

For each test case, print a single line containing the number of stupid submissions Abed has made.

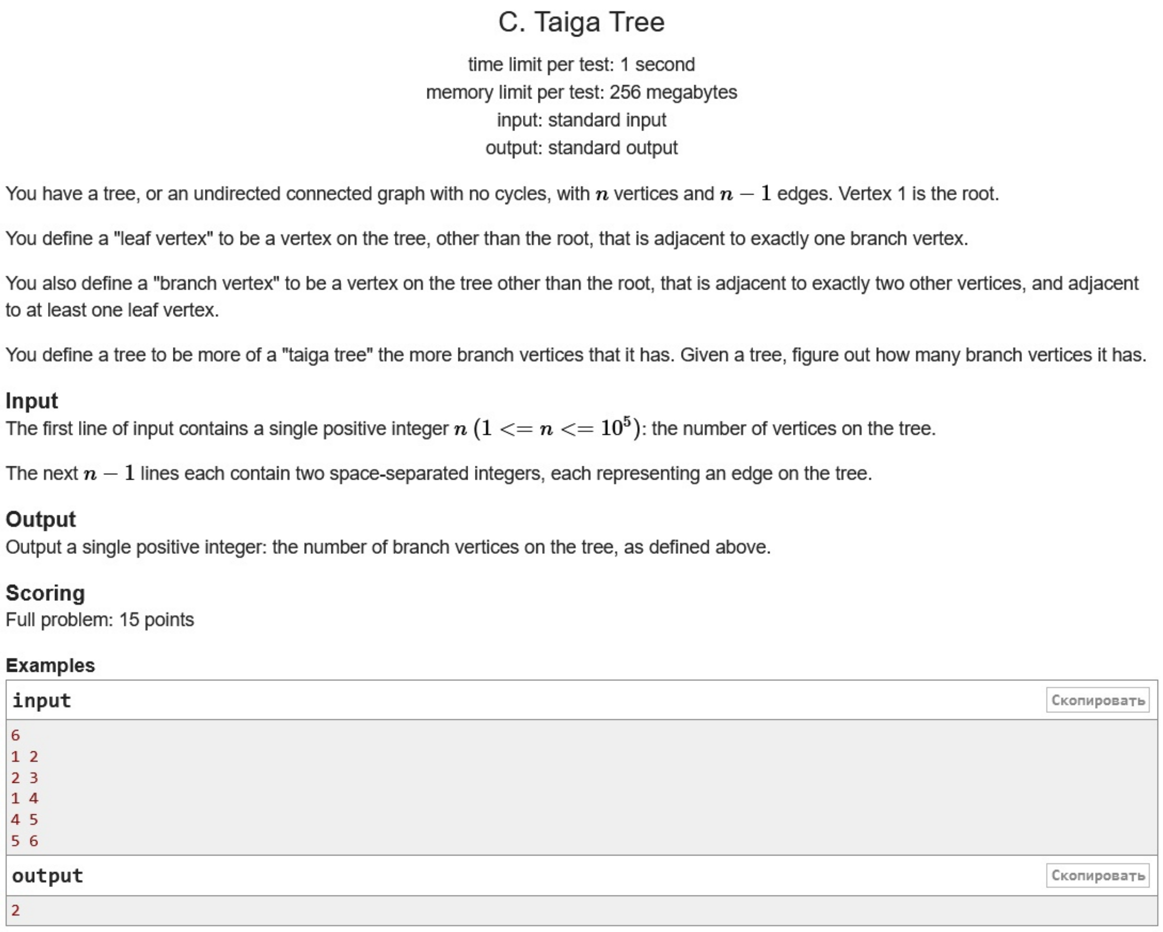
Решение

```
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

int main() {
    long long t;
        cin >> t;
    vector <char> v;

    long long n, q, res;
        int k;
        while (t--) {
                cin >> n >> q >> k;
                v.resize(n + 1);
                for (long long i = 1; i <= n; i++) {
                        cin >> v[i];
                }

                char c;
                res = 0;
                while (q--) {
                        cin >> c;
                        if (c == 'A') {
                                k = n+1;
                                continue;
                        } else {
                                int x;
                                cin >> x;
                                if (v[x] == 'S' && x <= k) {
                    res++;
                }
                                k = max(k, x);
                        }
                }
                cout << res << endl;
        }

}
```

Своя тренировка 3: Graphs



24	MAI #44 Vakhramyan, Kosogorov, Simonov	1	340			+7 03:20				-1			-10	
25	MAI #8 Grinberg, Karpov, Sankov	0	0			-11								
25	MAI #28 Batyanovsky, Veprintsev, Tyuneev	0	0			-9								
25	MAI #48 Burkevich, Kochuikova, Shukhova	0	0										-3	
25	MAI #30 Dergach, Moskalenko, Khasanov	0	0			-3						-3		-5
	* MAI #17 Belousov, Korotkevich, Sazhenov	3						+1			+2	+5		
	* MAI #41 Bronnikov, Veltman, Zhivykh	2							+3					+
	* MAI #20 Barsov, Nogaev, Petrosyan	1							+					
	* MAI #37 Efimov, Katermin	1											-2	+
	* MAI #24 Bakharev, Sindyukov	1											+	
	* MAI #44 Vakhramyan, Kosogorov, Simonov	1												+



C. Taiga Tree

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You have a tree, or an undirected connected graph with no cycles, with n vertices and $n-1$ edges. Vertex 1 is the root.

You define a "leaf vertex" to be a vertex on the tree, other than the root, that is adjacent to exactly one branch vertex.

You also define a "branch vertex" to be a vertex on the tree other than the root, that is adjacent to exactly two other vertices, and adjacent to at least one leaf vertex.

You define a tree to be more of a "taiga tree" the more branch vertices that it has. Given a tree, figure out how many branch vertices it has.

Input
The first line of input contains a single positive integer n $(1 <= n <= 10^5)$: the number of vertices on the tree.

The next $n-1$ lines each contain two space-separated integers, each representing an edge on the tree.

Output
Output a single positive integer: the number of branch vertices on the tree, as defined above.

Scoring
Full problem: 15 points

Examples

input Скопировать

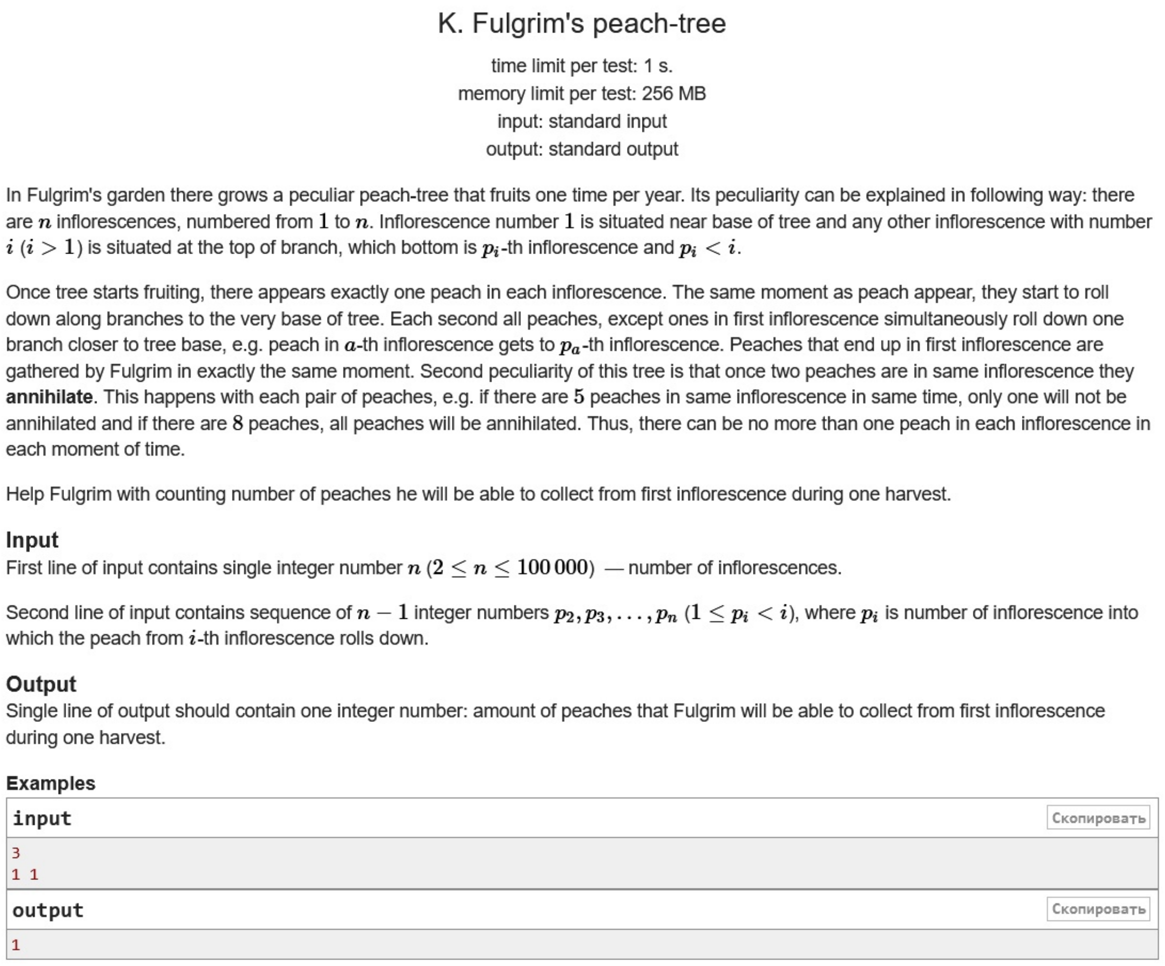
```
6
1 2
2 3
1 4
4 5
5 6
```

output Скопировать

```
2
```

Решение

```
#include <bits/stdc++.h>
using namespace std;

int main() {
  int n;
  std::cin >> n;
  if (n == 1 || n == 2)
    return 0;
  std::unordered_map<int, std::vector<int> > mapa;

  std::vector<int> c;
  c.assign(n + 2, 0);
  for (int i = 0; i < n - 1; ++i) {
    int a, b;
    std::cin >> a >> b;
    mapa[a].push_back(b);
    mapa[b].push_back(a);
    ++c[a];
    ++c[b];
  }
  std::vector<int> l;
  for (int i = 0; i < c.size(); ++i) {
    if (c[i] == 1 && i != 1) {
      l.push_back(i);
    }
  }
  int cnt = 0;
  for (int i = 0; i < l.size(); ++i) {
    if (mapa[l[i]][0] != 1 && c[mapa[l[i]][0]] == 2) {
      ++cnt;
    }
  }
  std::cout << cnt;
}
```



K. Fulgrim's peach-tree

time limit per test: 1 s.
memory limit per test: 256 MB
input: standard input
output: standard output

In Fulgrim's garden there grows a peculiar peach-tree that fruits one time per year. Its peculiarity can be explained in following way: there are n inflorescences, numbered from 1 to n. Inflorescence number 1 is situated near base of tree and any other inflorescence with number i ($i > 1$) is situated at the top of branch, which bottom is p_i-th inflorescence and $p_i < i$.

Once tree starts fruiting, there appears exactly one peach in each inflorescence. The same moment as peach appear, they start to roll down along branches to the very base of tree. Each second all peaches, except ones in first inflorescence simultaneously roll down one branch closer to tree base, e.g. peach in a-th inflorescence gets to p_a-th inflorescence. Peaches that end up in first inflorescence are gathered by Fulgrim in exactly the same moment. Second peculiarity of this tree is that once two peaches are in same inflorescence they **annihilate**. This happens with each pair of peaches, e.g. if there are 5 peaches in same inflorescence in same time, only one will not be annihilated and if there are 8 peaches, all peaches will be annihilated. Thus, there can be no more than one peach in each inflorescence in each moment of time.

Help Fulgrim with counting number of peaches he will be able to collect from first inflorescence during one harvest.

Input
First line of input contains single integer number n ($2 \le n \le 100\,000$) — number of inflorescences.

Second line of input contains sequence of $n-1$ integer numbers $p_2, p_3, \ldots, p_n$ ($1 \le p_i < i$), where p_i is number of inflorescence into which the peach from i-th inflorescence rolls down.

Output
Single line of output should contain one integer number: amount of peaches that Fulgrim will be able to collect from first inflorescence during one harvest.

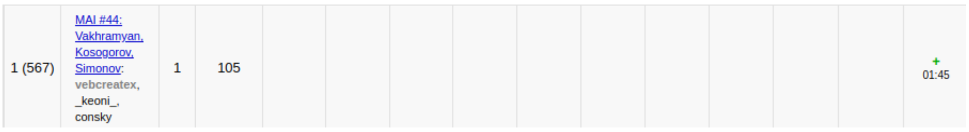
Examples

input Скопировать
```
3
1 1
```
output Скопировать
```
1
```

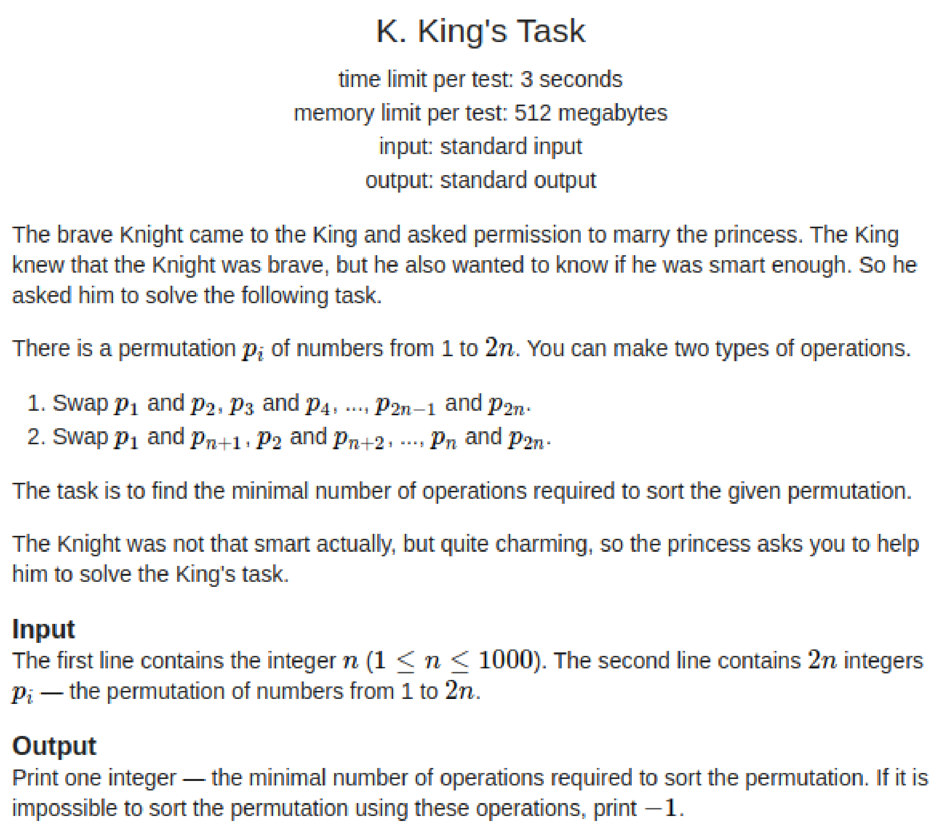
Решение

```
#include <iostream>
#include <vector>
#include <queue>

using namespace std;

int bfs(const vector<vector<int> >& graph) {
    queue<pair<int, int> > queue;
    vector<char> visited(graph.size(), 0);
    vector<int> depths(graph.size());
    queue.push({0, 1});

    while (!queue.empty()) {
        pair<int, int> fr = queue.front();
        queue.pop();
        ++depths[fr.second - 1];

        const vector<int>& edge = graph[fr.first];
        for (const auto& node : edge) {
            queue.push({node, fr.second + 1});
        }
    }

    int res = 0;
    for (const auto& e : depths) {
        res += e % 2;
    }

    return res;
}



int main() {
    size_t n;
    cin >> n;
    vector<vector<int> > graph(n);

    for (size_t i = 1; i < n; ++i) {
        size_t p;
        cin >> p;
        graph[p - 1].push_back(i);
    }
    cout << bfs(graph) << '\n';
}
```

2020-2021 ICPC, NERC, Northern Eurasia Onsite



1 (567)	MAI #44: Vakhramyan, Kosogorov, Simonov: vebcreatex, _keoni_, consky	1	105											+ 01:45



K. King's Task

time limit per test: 3 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

The brave Knight came to the King and asked permission to marry the princess. The King knew that the Knight was brave, but he also wanted to know if he was smart enough. So he asked him to solve the following task.

There is a permutation p_i of numbers from 1 to $2n$. You can make two types of operations.

1. Swap p_1 and p_2, p_3 and p_4, ..., p_{2n-1} and p_{2n}.
2. Swap p_1 and p_{n+1}, p_2 and p_{n+2}, ..., p_n and p_{2n}.

The task is to find the minimal number of operations required to sort the given permutation.

The Knight was not that smart actually, but quite charming, so the princess asks you to help him to solve the King's task.

Input
The first line contains the integer n ($1 \le n \le 1000$). The second line contains $2n$ integers p_i — the permutation of numbers from 1 to $2n$.

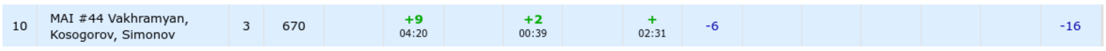
Output
Print one integer — the minimal number of operations required to sort the permutation. If it is impossible to sort the permutation using these operations, print -1.

Решение

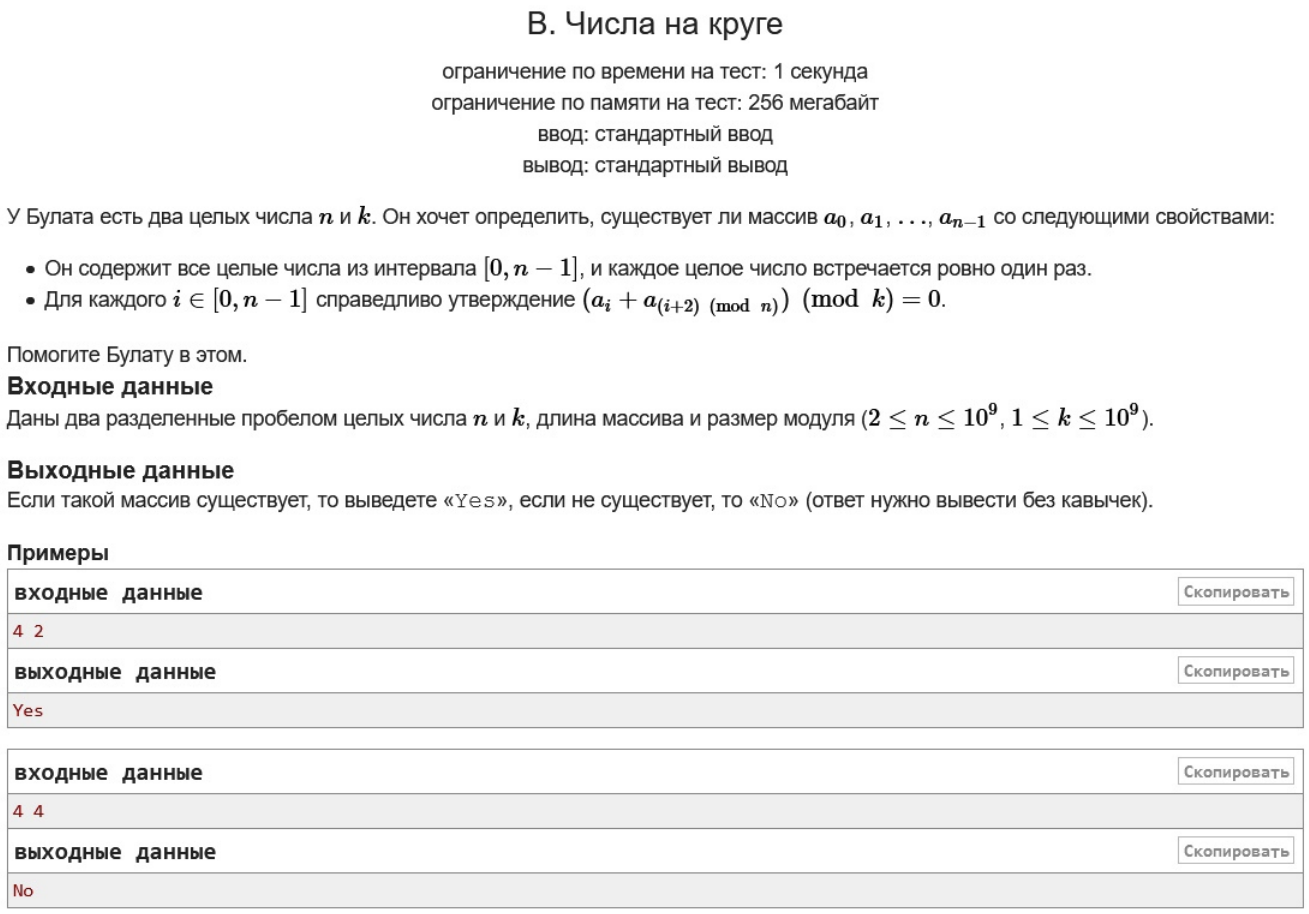
```
#include <bits/stdc++.h>
#define fast ios_base::sync_with_stdio(false); cin.tie(NULL); cout.tie(NULL)
#define flush cout.flush()

 using namespace std;

 vector<int> oper1(vector<int> v1) {
     vector<int> v = v1;
     size_t n = v.size();
     for (size_t i = 0; i < n - 1; i += 2)
         swap(v[i], v[i + 1]);
```

```
    return v;
}

vector<int> oper2(vector<int> v1) {
    vector<int> v = v1;
    int n = v.size();
    for (size_t i = 0; i < n / 2; i++)
        swap(v[i], v[(n / 2) + i]);

    return v;
}


int solve(vector<int>& v, int flag, int count) {
    if (count > v.size()) {
        return -1;
    }
    if (is_sorted(v.begin(), v.end())) {

        return count;
    }
    vector<int> tmp;
    count++;
    if (flag == 0) {
        tmp = oper1(v);
        flag = 1;
        return solve(tmp, flag, count);
    } else {
        tmp = oper2(v);
        flag = 0;
        return solve(tmp, flag, count);
    }
}







int main() {
    fast;

    int n;
    cin >> n;
    vector<int> v(2 * n);
    for (auto & a : v)
        cin >> a;

    if (is_sorted(v.begin(), v.end())) {
        cout << "0\n";
        return 0;
    }


    vector<int> tmp = oper1(v);
    int count = 1;
    int ans1 = solve(tmp, 1, count);
    tmp = oper2(v);
    count = 1;
    int ans2 = solve(tmp, 0, count);

    if (ans1 == -1 and ans2 == -1)
        cout << "-1\n";
    else {
        if (ans1 < ans2 and ans1 != -1)
            cout << ans1 << '\n';
        else if (ans2 <= ans1 and ans2 != -1)
            cout << ans2 << '\n';
    }

    return 0;
}
```

20-21 Своя тренировка 4



10	MAI #44 Vakhramyan, Kosogorov, Simonov	3	670		+9 04:20		+2 00:39		+ 02:31	-6						-16



B. Числа на круге

ограничение по времени на тест: 1 секунда
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

У Булата есть два целых числа n и k. Он хочет определить, существует ли массив $a_0, a_1, \ldots, a_{n-1}$ со следующими свойствами:

- Он содержит все целые числа из интервала $[0, n-1]$, и каждое целое число встречается ровно один раз.
- Для каждого $i \in [0, n-1]$ справедливо утверждение $(a_i + a_{(i+2) \pmod n}) \pmod k = 0$.

Помогите Булату в этом.

Входные данные

Даны два разделенные пробелом целых числа n и k, длина массива и размер модуля ($2 \le n \le 10^9$, $1 \le k \le 10^9$).

Выходные данные

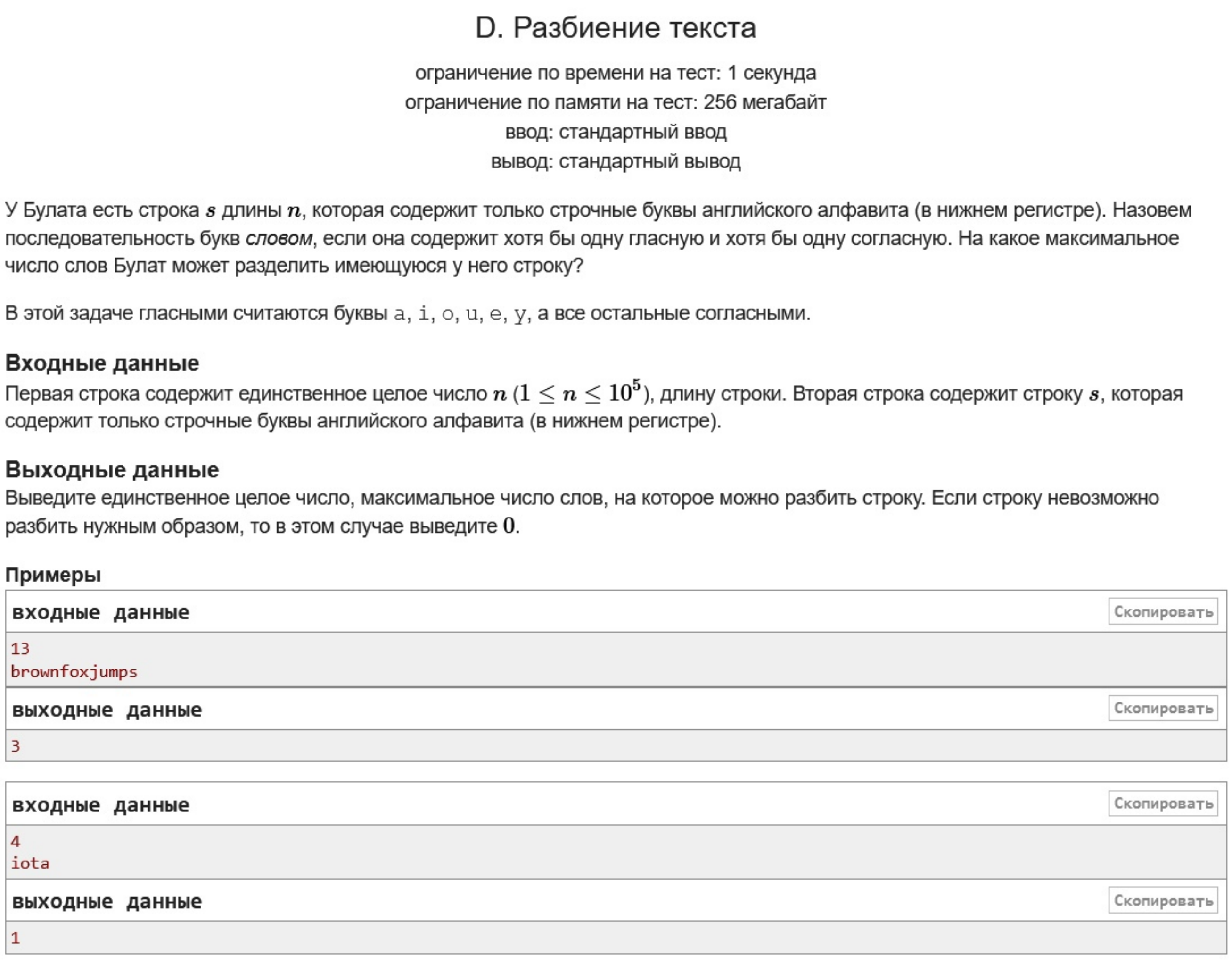
Если такой массив существует, то выведете «Yes», если не существует, то «No» (ответ нужно вывести без кавычек).

Примеры

входные данные Скопировать
```
4 2
```
выходные данные Скопировать
```
Yes
```
входные данные Скопировать
```
4 4
```
выходные данные Скопировать
```
No
```

Решение

```
#include <iostream>

using namespace std;

int main() {
    size_t n, k;
    cin >> n >> k;
    if (k == 1 or (n % 2 == 0 and k == 2) or (n == 4 and k == 3)) {
        cout << "Yes\n";
        return 0;
    }
    cout << "No\n";
}
```



D. Разбиение текста

ограничение по времени на тест: 1 секунда
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

У Булата есть строка s длины n, которая содержит только строчные буквы английского алфавита (в нижнем регистре). Назовем последовательность букв *словом*, если она содержит хотя бы одну гласную и хотя бы одну согласную. На какое максимальное число слов Булат может разделить имеющуюся у него строку?

В этой задаче гласными считаются буквы a, i, o, u, e, y, а все остальные согласными.

Входные данные

Первая строка содержит единственное целое число n ($1 \leq n \leq 10^5$), длину строки. Вторая строка содержит строку s, которая содержит только строчные буквы английского алфавита (в нижнем регистре).

Выходные данные

Выведите единственное целое число, максимальное число слов, на которое можно разбить строку. Если строку невозможно разбить нужным образом, то в этом случае выведите 0.

Примеры

входные данные

```
13
brownfoxjumps
```

выходные данные

```
3
```

входные данные

```
4
iota
```

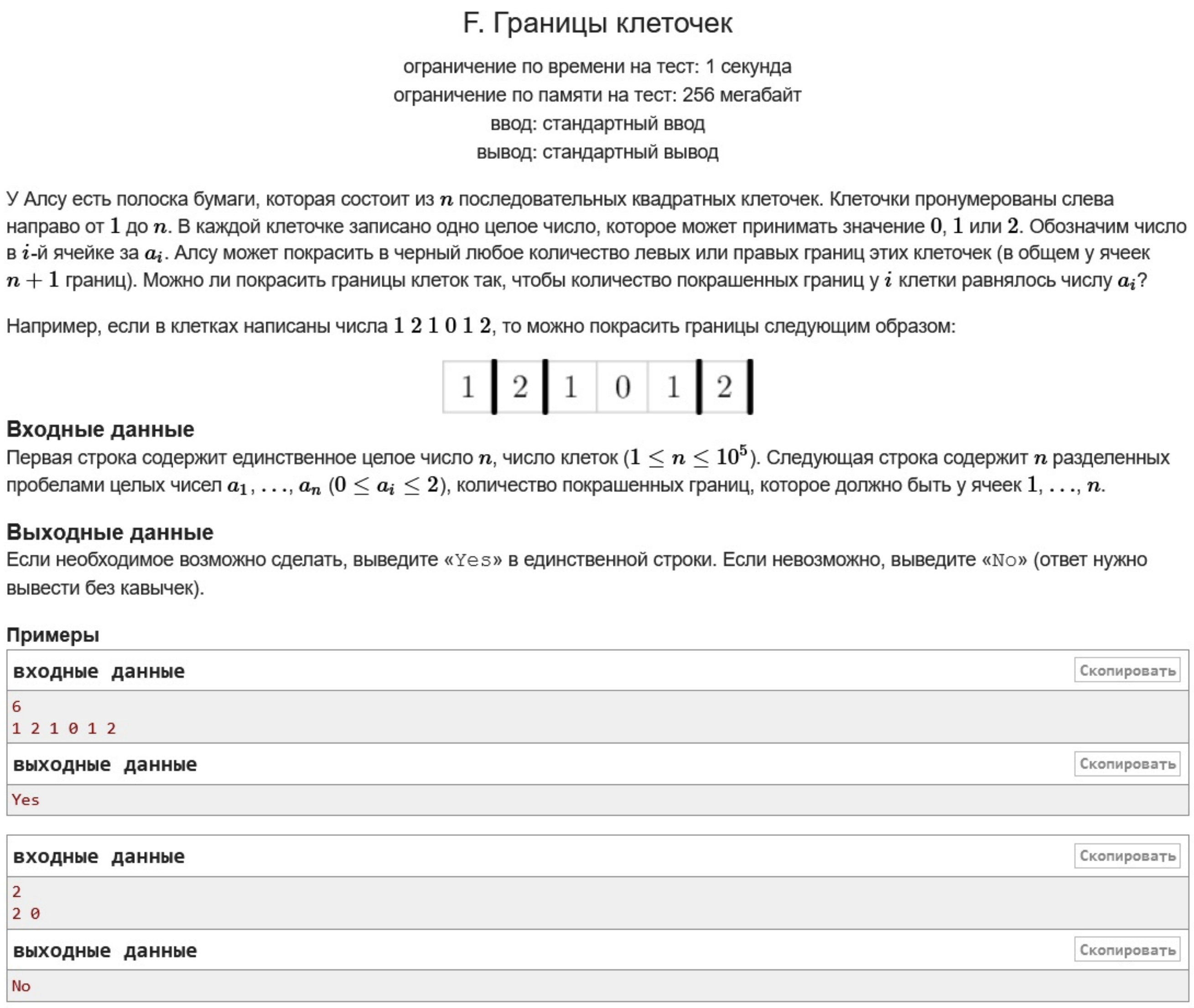
выходные данные

```
1
```

Решение

```
#include <vector>
#include <string>
#include <iostream>
using namespace std;

int main() {
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);
    cout.tie(NULL);

    int n;
    cin >> n;
```

```
vector<int> v;
int input;
for (int i = 0; i < n; ++i) {
    cin >> input;
    v.push_back(input);
}

vector<int> bars(n + 1, 1);
bool res = true;
for (int i = 0; i < n; ++i) {
    if (v[i] == 2) {
        bars[i] = 2;
        bars[i + 1] = 2;
    } else if (v[i] == 0) {
        bars[i] = 0;
        bars[i + 1] = 0;
    }
}

for (int i = 0; i < n; ++i) {
    if (v[i] == 1) {
        if (bars[i] == 2) {
            bars[i + 1] = 0;
        } else if (bars[i + 1] == 2) {
            bars[i] = 0;
        } else if (bars[i] == 0) {
            bars[i + 1] = 2;
        } else if (bars[i + 1] == 0) {
            bars[i] = 2;
        } else {
            int count = 0;
            int last = 1;
            int j = i;
            while (j < n && v[j] == 1) {
                ++count;
                ++j;
            }
            if (j != n) {
                last = v[j];
            }
            if (last == 1) {
                bars[i + 1] = 2;
                bars[i] = 0;
            } else if (last == 2 && count % 2 == 0) {
                    bars[i] = 2;
                    bars[i + 1] = 0;
                } else if (last == 2 && count % 2 != 0) {
                    bars[i + 1] = 2;
                    bars[i] = 0;
                } else if (last == 0 && count % 2 == 0) {
                    bars[i] = 0;
                    bars[i + 1] = 2;
                } else if (last == 0 && count % 2 != 0) {
                    bars[i + 1] = 0;
                    bars[i] = 2;
                }
            }
        }
    }

    for (int i = 0; i < n; ++i) {
        if (v[i] == 0 && (bars[i] == 2 || bars[i + 1] == 2)) {
            res = false;
        } else if (v[i] == 2 && !(bars[i] == 2 && bars[i + 1] == 2)) {
            res = false;
        } else if (v[i] == 1 && !((bars[i] == 2 && bars[i + 1] == 0) ||
            bars[i] == 0 && bars[i + 1] == 2)) {
            res = false;
        }
    }

    if (res) {
        cout << "Yes" << endl;
        return 0;
    }
    cout << "No" << endl;
}
```

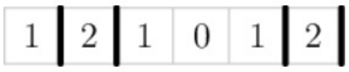


F. Границы клеточек

ограничение по времени на тест: 1 секунда
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

У Алсу есть полоска бумаги, которая состоит из n последовательных квадратных клеточек. Клеточки пронумерованы слева направо от 1 до n. В каждой клеточке записано одно целое число, которое может принимать значение 0, 1 или 2. Обозначим число в i-й ячейке за a_i. Алсу может покрасить в черный любое количество левых или правых границ этих клеточек (в общем у ячеек $n+1$ границ). Можно ли покрасить границы клеток так, чтобы количество покрашенных границ у i клетки равнялось числу a_i?

Например, если в клетках написаны числа $1\ 2\ 1\ 0\ 1\ 2$, то можно покрасить границы следующим образом:



Входные данные
Первая строка содержит единственное целое число n, число клеток ($1 \le n \le 10^5$). Следующая строка содержит n разделенных пробелами целых чисел $a_1, \ldots, a_n$ ($0 \le a_i \le 2$), количество покрашенных границ, которое должно быть у ячеек $1, \ldots, n$.

Выходные данные
Если необходимое возможно сделать, выведите «Yes» в единственной строки. Если невозможно, выведите «No» (ответ нужно вывести без кавычек).

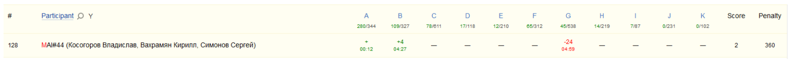
Примеры

входные данные Скопировать
```
6
1 2 1 0 1 2
```
выходные данные Скопировать
```
Yes
```
входные данные Скопировать
```
2
2 0
```
выходные данные Скопировать
```
No
```

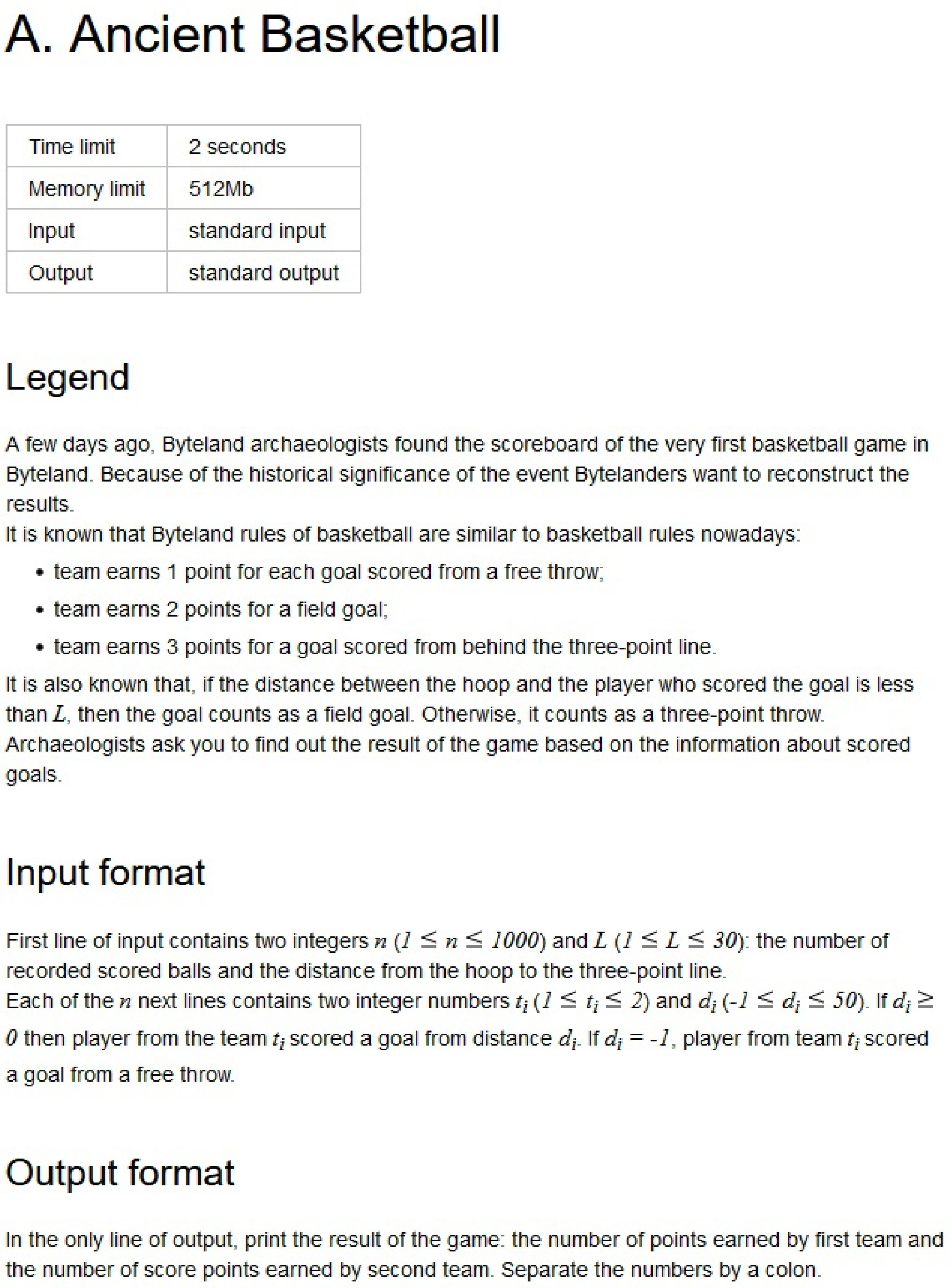
Решение

```
#include <iostream>
#include <string>
#include <vector>

using namespace std;

int main() {
    int n;
    cin >> n;
    string s;
    cin >> s;
    vector<char> vowels = {'a', 'i', 'o', 'u', 'e', 'y'};

    size_t vowel_count = 0;
    size_t consonant_count = 0;
    size_t res = 0;

    for (auto c : s) {
        bool got_vowel = false;
        for (auto it = vowels.begin(); it != vowels.end(); it++) {
            if (*it == c) {
                vowel_count++;
                got_vowel = true;
                break;
            }
        }
        if (got_vowel == false) {
            consonant_count++;
        }
        if (consonant_count > 0 && vowel_count > 0) {
            res++;
            consonant_count = 0;
            vowel_count = 0;
        }
    }
    cout << res << endl;
}
```

RuCode Spring 2021 Championship C-E



#	Participant	A 280/344	B 109/327	C 78/611	D 17/118	E 12/210	F 65/312	G 45/538	H 14/219	I 7/87	J 0/231	K 0/102	Score	Penalty
128	MAI#44 (Косогоров Владислав, Вахрамян Кирилл, Симонов Сергей)	+ 00:12	+4 04:27	—	—	—	—	-24 04:59	—	—	—	—	2	360



A. Ancient Basketball

Time limit	2 seconds
Memory limit	512Mb
Input	standard input
Output	standard output

Legend

A few days ago, Byteland archaeologists found the scoreboard of the very first basketball game in Byteland. Because of the historical significance of the event Bytelanders want to reconstruct the results.
It is known that Byteland rules of basketball are similar to basketball rules nowadays:

- team earns 1 point for each goal scored from a free throw;
- team earns 2 points for a field goal;
- team earns 3 points for a goal scored from behind the three-point line.

It is also known that, if the distance between the hoop and the player who scored the goal is less than L, then the goal counts as a field goal. Otherwise, it counts as a three-point throw.
Archaeologists ask you to find out the result of the game based on the information about scored goals.

Input format

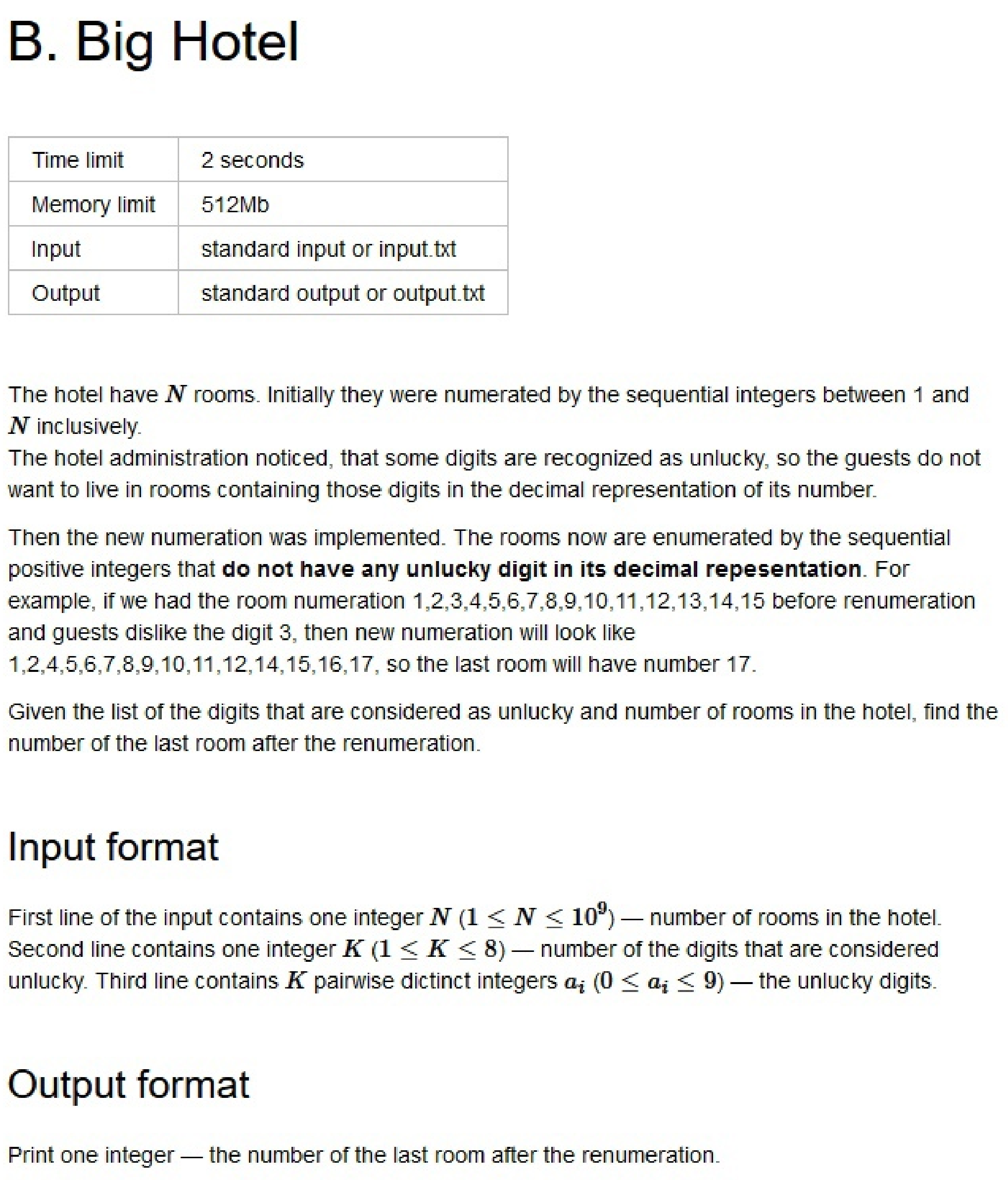
First line of input contains two integers n ($1 \le n \le 1000$) and L ($1 \le L \le 30$): the number of recorded scored balls and the distance from the hoop to the three-point line.
Each of the n next lines contains two integer numbers t_i ($1 \le t_i \le 2$) and d_i ($-1 \le d_i \le 50$). If $d_i \ge 0$ then player from the team t_i scored a goal from distance d_i. If $d_i = -1$, player from team t_i scored a goal from a free throw.

Output format

In the only line of output, print the result of the game: the number of points earned by first team and the number of score points earned by second team. Separate the numbers by a colon.

Решение

```
#include <bits/stdc++.h>
#define fast ios_base::sync_with_stdio(false); cin.tie(NULL); cout.tie(NULL)
#define ld long double
#define ll long long
#define ull unsigned long long

using namespace std;

int main() {
    fast;
    int n, l;
    cin >> n >> l;
    int p1 = 0, p2 = 0;
    while (n--) {
        int t, d;
        cin >> t >> d;
        if (t == 1) {
            if (d == -1)
                p1++;
            else if (d < l)
                p1 += 2;
            else
                p1 += 3;
        }
        else {
            if (d == -1)
                p2++;
            else if (d < l)
                p2 += 2;
            else
                p2 += 3;
        }
    }
    cout << p1 << ':' << p2 <<'\n';
    return 0;
}
```



B. Big Hotel

Time limit	2 seconds
Memory limit	512Mb
Input	standard input or input.txt
Output	standard output or output.txt

The hotel have N rooms. Initially they were numerated by the sequential integers between 1 and N inclusively.
The hotel administration noticed, that some digits are recognized as unlucky, so the guests do not want to live in rooms containing those digits in the decimal representation of its number.

Then the new numeration was implemented. The rooms now are enumerated by the sequential positive integers that **do not have any unlucky digit in its decimal repesentation**. For example, if we had the room numeration 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 before renumeration and guests dislike the digit 3, then new numeration will look like 1,2,4,5,6,7,8,9,10,11,12,14,15,16,17, so the last room will have number 17.

Given the list of the digits that are considered as unlucky and number of rooms in the hotel, find the number of the last room after the renumeration.

Input format

First line of the input contains one integer N $(1 \leq N \leq 10^9)$ — number of rooms in the hotel. Second line contains one integer K $(1 \leq K \leq 8)$ — number of the digits that are considered unlucky. Third line contains K pairwise dictinct integers a_i $(0 \leq a_i \leq 9)$ — the unlucky digits.

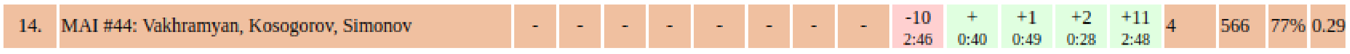
Output format

Print one integer — the number of the last room after the renumeration.

Решение

```
n, k = int(input()), int(input())

s = ""
unwanted = [int(i) for i in input().split()]
nums = [i for i in range(10) if i not in unwanted]

while n > 0:
    s = str(n % (10 - k)) + s
    n //= (10 - k)
print(''.join([str(nums[int(s[i])]) for i in range(len(s))]))
```

XXI Open Cup named after E.V. Pankratiev. Grand prix of China, Division 2



14.	MAI #44: Vakhramyan, Kosogorov, Simonov	-	-	-	-	-	-	-	-	-10 2:46	+ 0:40	+1 0:49	+2 0:28	+11 2:48	4	566	77%	0.29

Problem O. Juggle

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

You are performing a magic trick with a special deck of cards.

You lay out the cards in a row from left to right, face up. Each card has a lower-case letter on it. Two cards with the same letter are indistinguishable. You select an audience member to perform an operation on the cards. You will not see what operation they perform.

The audience member can do one of two thingsthey can either select any two cards and swap them, or leave the cards untouched.

In order for the trick to succeed, you must correctly guess what the audience member dideither you guess that the audience member did nothing, or you point at the two cards the audience member swapped.

Given a string that represents the initial arrangement of the cards, can you guarantee that you will always be able to guess the audience member's operation correctly, no matter what operation they perform?

Input

The input consists of a single line containing the string s ($1 \le |s| \le 50$), which represents the initial arrangement of the cards, in the order they appear in the row. The string contains only lower-case letters ('a'-'z').

Output

Output a single line with 1 if you can guarantee that you will always be able to guess the audience member's operation correctly, or 0 otherwise.

Решение

```
#include <bits/stdc++.h>
#define fast ios_base::sync_with_stdio(false); cin.tie(NULL); cout.tie(NULL)
#define ld long double
#define ll long long
#define ull unsigned long long

using namespace std;

int main() {
    fast;
    string str;
    cin >> str;
    unordered_map<char, int> m;
    for (size_t i = 0; i != str.length(); i++) {
        m[str[i]]++;
        if (m[str[i]] > 1) {
            cout << "0\n";
            return 0;
        }
    }
    cout << 1 << '\n';

    return 0;
}
```

Problem P. Kindergarten

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

You are teaching kindergarten! You wrote down the numbers from 1 to n, in order, on a whiteboard. When you weren't paying attention, one of your students erased one of the numbers.

Can you tell which number your mischievous student erased?

Input

The first line of input contains a single integer n ($2 \le n \le 100$), which is the number of numbers that you wrote down. The second line of input contains a string of digits, which represents the numbers you wrote down (minus the one that has been erased). There are no spaces in this string. It is guaranteed to contain all of the numbers from 1 to n, in order, except for the single number that the student erased.

Output

Output a single integer, which is the number that the tricky student erased.

Решение

```
n = int(input())
s = input()



nums = [str(i) for i in range(1, n+1)]
cur_idx = 0
for num in nums:
    if s[cur_idx:cur_idx+len(num)] != num:
        print(num)
        break
    cur_idx += len(num)
```

Problem Q. Minimum and Maximum Ratings

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

Your judges are preparing a problem set, and they're trying to evaluate a problem for inclusion in the set. Each judge rates the problem with an integer between -3 and 3.

The overall rating of the problem is the average of all of the judges' ratings — that is, the sum of the ratings divided by the number of judges providing a rating.

Some judges have already rated the problem. Compute the minimum and maximum possible overall rating that the problem can end up with after the other judges submit their ratings.

Input

The first line of input contains two integers n ($1 \leq n \leq 10$) and k ($0 \leq k \leq n$), where n is the total number of judges, and k is the number of judges who have already rated the problem. Each of the next k lines contains a single integer r ($-3 \leq r \leq 3$). These are the ratings of the k judges that have already rated the problem.

Output

Output two space-separated floating point numbers on a single line, which are the minimum and maximum overall rating the problem could achieve after the remaining judges rate the problem, minimum first. These values must be accurate to an absolute or relative error of 10^{-4}.

Решение

```
#include <bits/stdc++.h>
#define fast ios_base::sync_with_stdio(false); cin.tie(NULL); cout.tie(NULL)
#define ld long double
#define ll long long
#define ull unsigned long long

using namespace std;

int main() {
    fast;


    int n, k;
    cin >> n >> k;
    vector<int> v(k);
    int sum = 0;

    for (auto& a : v) {
        cin >> a;
        sum += a;
    }
    double min = (sum + (n - k) * (-3)) / double(n);
    double max = (sum + (n - k) * 3) / double(n);

    cout << setprecision (18) << fixed << min << ' ' << max;



    return 0;
}
```

Problem R. Public Mails

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

There is a group of people in an internet email message group. Messages are sent to all members of the group, and no two messages are sent at the same time.

Immediately before a person sends a message, they read all their unread messages up to that point. Each sender also reads their own message the moment it is sent. Therefore, a person's unread messages are exactly the set of messages sent after that person's last message.

Each time a message is sent, compute the total number of unread messages over all group members.

Input

The first line of input contains two integers n ($1 \le n \le 10^9$) and m ($1 \le m \le 1{,}000$), where n is the number of people in the group, and m is the number of messages sent. The group members are identified by number, 1 through n.

Each of the next m lines contains a single integer s ($1 \le s \le n$), which is the sender of that message. These lines are in chronological order.

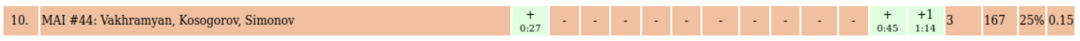
Output

Output m lines, each with a single integer, indicating the total number of unread messages over all group members, immediately after each message is sent.

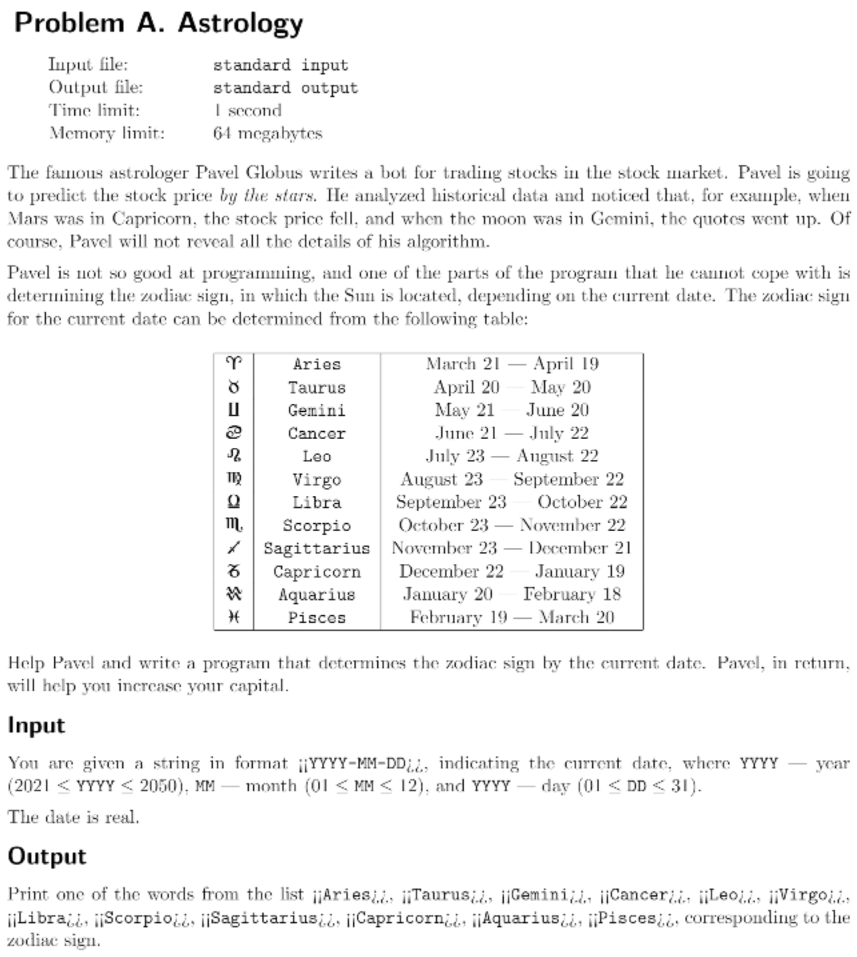
Решение

```
n, m = [int(i) for i in input().split()]
res = 0
sent = []

for i in range(m):
        person = int(input())
        unread = 0

        if (len(sent) > 0):
                if (person in sent):
                        sent = list(reversed(sent))
                        unread = sent.index(person)
                        sent = list(reversed(sent))
                else:
                        unread = len(sent)
        sent.append(person)

        res -= unread
        res += n - 1

        print (res)
```

XXI Open Cup named after E.V. Pankratiev. Grand Prix of Urals, Division 2



10.	MAI #44: Vakhramyan, Kosogorov, Simonov	+ 0:27	-	-	-	-	-	-	-	-	-	-	+ 0:45	+1 1:14	3	167	25%	0.15



Problem A. Astrology

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	64 megabytes

The famous astrologer Pavel Globus writes a bot for trading stocks in the stock market. Pavel is going to predict the stock price *by the stars*. He analyzed historical data and noticed that, for example, when Mars was in Capricorn, the stock price fell, and when the moon was in Gemini, the quotes went up. Of course, Pavel will not reveal all the details of his algorithm.

Pavel is not so good at programming, and one of the parts of the program that he cannot cope with is determining the zodiac sign, in which the Sun is located, depending on the current date. The zodiac sign for the current date can be determined from the following table:

♈	`Aries`	March 21 — April 19
♉	`Taurus`	April 20 — May 20
♊	`Gemini`	May 21 — June 20
♋	`Cancer`	June 21 — July 22
♌	`Leo`	July 23 — August 22
♍	`Virgo`	August 23 — September 22
♎	`Libra`	September 23 — October 22
♏	`Scorpio`	October 23 — November 22
♐	`Sagittarius`	November 23 — December 21
♑	`Capricorn`	December 22 — January 19
♒	`Aquarius`	January 20 — February 18
♓	`Pisces`	February 19 — March 20

Help Pavel and write a program that determines the zodiac sign by the current date. Pavel, in return, will help you increase your capital.

Input

You are given a string in format ¡¡`YYYY-MM-DD`¿¿, indicating the current date, where `YYYY` — year ($2021 \le$ `YYYY` ≤ 2050), `MM` — month ($01 \le$ `MM` ≤ 12), and `YYYY` — day ($01 \le$ `DD` ≤ 31).

The date is real.

Output

Print one of the words from the list ¡¡`Aries`¿¿, ¡¡`Taurus`¿¿, ¡¡`Gemini`¿¿, ¡¡`Cancer`¿¿, ¡¡`Leo`¿¿, ¡¡`Virgo`¿¿, ¡¡`Libra`¿¿, ¡¡`Scorpio`¿¿, ¡¡`Sagittarius`¿¿, ¡¡`Capricorn`¿¿, ¡¡`Aquarius`¿¿, ¡¡`Pisces`¿¿, corresponding to the zodiac sign.

Решение

```
year, month, day = [int(i) for i in input().split('-')]

if (month == 3 and day in range(21, 32)) or
    (month == 4 and day in range(1, 20)):
    print('Aries')
elif (month == 4 and day in range(20, 21)) or
    (month == 5 and day in range(1, 21)):
    print('Taurus')
elif (month == 5 and day in range(21, 32)) or
    (month == 6 and day in range(1, 21)):
    print('Gemini')
elif (month == 6) or (month == 7 and day in range(1, 23)):
    print('Cancer')
elif (month == 7) or (month == 8 and day in range(1, 23)):
    print('Leo')
elif (month == 8) or (month == 9 and day in range(1, 23)):
    print('Virgo')
elif (month == 9) or (month == 10 and day in range(1, 23)):
    print('Libra')
elif (month == 10) or (month == 11 and day in range(1, 23)):
    print('Scorpio')
elif (month == 11) or (month == 12 and day in range(1, 22)):
    print('Sagittarius')
elif (month == 12) or (month == 1 and day in range(1, 20)):
    print('Capricorn')
elif (month == 1) or (month == 2 and day in range(1, 19)):
    print('Aquarius')
elif (month == 2) or (month == 3 and day in range(1, 21)):
    print('Pisces')
```

Problem N. Numbers

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	64 megabytes

A non-negative integer is K-digit, if it can be written using K digits, but $(K-1)$ digits are not enough for this. For example, 43 is a two-digit number, 2010 is four-digit, and 0 and 5 are one-digit numbers.

For given A and B count, how many K-digit non-negative integers exist, where K is not less than A and not greater than B.

Input

The only line contains integers A and B ($1 \le A \le B \le 1000$).

Output

Output the answer to the problem without excessive leading zeros.

Решение

```
a, b = map(int, input().split())
ans = 0

for i in range(a, b+1):
    ans += 10**i
    ans -= 10**(i-1)
    if i == 1:
        ans += 1

print(ans)
```

Problem O. Overwhelming Vowels

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	64 megabytes

There are 26 letters in English alphabet, of which 5 (‘a’, ‘e’, ‘i’, ‘o’, ‘u’) are vowels, 20 are consonants and one is considered a semivowel, i.e. neither a vowel nor a consonant (letter ‘y’). A word is sonorous if the number of vowels in it exceeds the number of consonants. If a word contains some letter more than once, then each its occurence counts — for example, “alabama” is a sonorous word.

For a given word determine the least number of letters in it which must be replaced to obtain a sonorous word.

Input

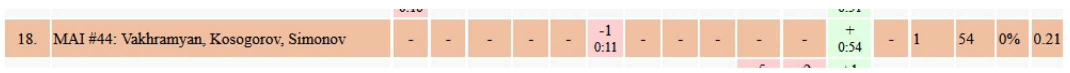
A single line of input contains the word given. The length of the given word does not exceed 1000 letters. The word consists of lowercase letters.

Output

Output the least number of letters replacement of which makes the given word sonorous.

Решение

```
word = input()
vowels = ['a', 'e', 'i', 'o', 'u']

vowel_count = 0
for letter in word:
        if letter in vowels:
                vowel_count += 1

print((len(word) - word.count('y')) // 2 + 1 - vowel_count)
```



18.	MAI #44: Vakhramyan, Kosogorov, Simonov	-	-	-	-	-	-1 0:11	-	-	-	-	-	+ 0:54	-	1	54	0%	0.21

Problem Q. Dancers

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 Mebibytes

Alex missed the ballroom dance competition that he wanted to attend. So now he wants to know the pairs of dancers whose dancing he missed. He had several photos from the competition, so he chose one where all dancers are clearly visible and wrote down the coordinates of all N dancers (N is even). Then Alex determined the pairs of dancers by the following algorithm: from not yet paired dancers he chooses two closest (to each other) dancers and assumes that they dance together as a pair. Should he find several pairs of dancers with the same minimum distance between dancers, he chooses lexicographically smallest pair (Alex enumerated dancers by integer numbers from 1 to N, dancers are ordered inside a pair, one with lower number goes first). You are asked to help Alex to determine pairs of dancers.

Input

The first line of input contains even integer N ($2 \le N \le 300$). Each i-th line of the next N lines contains two integers — x and y coordinates of point, representing i-th dancer. All points are distinct. All coordinates are less than 10^8 by absolute value.

Output

You should output $N/2$ lines. Each line must contain numbers of dancers in the corresponding pair. The first number in a line should be less than the second. Lines must be sorted in the lexicographically ascending order.

Решение

```
#include <iostream>
#include <map>
#include <cmath>
#include <algorithm>
#include <limits>
#include <vector>

using namespace std;

int main() {
    int N;
    cin >> N;

    map<int, pair<double, double> > skater;
    for (int i = 1; i <= N; ++i) {
        double x, y;
        cin >> x >> y;
        skater[i] = {x, y};
    }

    map<pair<int, int>, double > distances;
    for (int i = 1; i < N; ++i) {
        double dist;

        for (int j = i + 1; j <= N; ++j) {
            distances[{i, j}] = sqrt(pow(skater[i].first - skater[j].first, 2) +
        }

    }

    vector<pair<int, int> > ans;
    while (!distances.empty()) {
        double min_dist = numeric_limits<double>::max();
        for (auto elem : distances) {
            if (elem.second < min_dist) {
                min_dist = elem.second;
                // cout << min_dist << endl;
            }
        }

        vector<pair<int, int> > min_pairs;
        for (auto it = distances.begin(); it != distances.end(); ++it) {
            if (it->second == min_dist) {
                int a = it->first.first;
                int b = it->first.second;
                if (a <= b) {
                    min_pairs.push_back({a, b});
                } else {
                    min_pairs.push_back({b, a});
                }
            }
        }

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

        ans.push_back(min_pairs[0]);
        // cout << min_pairs[0].first << ' ' << min_pairs[0].second << endl;

        vector<pair<int, int> > to_erase;
        for (auto elem : distances) {
                if (elem.first.first == min_pairs[0].first ||
                    elem.first.first == min_pairs[0].second ||
                            elem.first.second == min_pairs[0].first ||
                            elem.first.second == min_pairs[0].second) {
                    to_erase.push_back({elem.first.first, elem.first.second});
                            }
            }
            min_pairs.clear();

            for (auto elem : to_erase) {
                distances.erase(elem);
            }

            to_erase.clear();
        }

        sort(ans.begin(), ans.end());
        for (auto elem : ans) {
            cout << elem.first << ' ' << elem.second << endl;
        }
}
```