A - Card Trick

I am learning magic tricks to impress my girlfriend Alice. My latest trick is a probabilistic one, i.e. it does work in most cases, but not in every case. To perform the trick, I first shuffle a set of many playing cards and put them all in one line with faces up on the table. Then Alice secretly selects one of the first ten cards (i.e. she chooses x_0, a secret number between 1 and 10 inclusive) and skips cards repeatedly as follows: after having selected a card at position x_i with a number $c(x_i)$ on its face, she will select the card at position $x_{i+1} = x_i + c(x_i)$. Jack (J), Queen (Q), and King (K) count as 10, Ace (A) counts as 11. You may assume that there are at least ten cards on the table.

Alice stops this procedure as soon as there is no card at position $x_i + c(x_i)$. I then perform the same procedure from a randomly selected starting position that may be different from the position selected by Alice. It turns out that often, I end up at the same position. Alice is very impressed by this trick.

However, I am more interested in the underlying math. Given my randomly selected starting position and the card faces of every selected card (including my final one), can you compute the probability that Alice chose a starting position ending up on the same final card? You may assume that her starting position is randomly chosen with uniform probability (between 1 and 10 inclusive). I forgot to note the cards that I skipped, so these cards are unknown. You may assume that the card face of every single of the unknown cards is independent of the other card faces and random with uniform probability out of the possible card faces (i.e. 2-10, J, Q, K, and A).

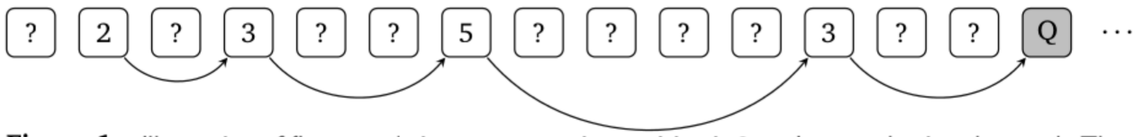


Figure 1 – Illustration of first sample input: my starting position is 2, so I start selecting that card. Then I keep skipping cards depending on the card's face. This process iterates until there are not enough cards to skip (in this sample: Q). The final Q card is followed by 0 to 9 unknown cards, since Q counts as 10.

Input

For each test case:

- A line containing two integers n ($1 \le n \le 100$) and m ($1 \le m \le 10$) where n is the number of selected cards and m is the 1-based position of my first selected card.
- A line with n tokens that specify the n selected card faces (in order, including the final card). Each card face is given either as an integer x ($2 \le x \le 10$) or as a single character (J, Q, K, or A as specified above).

Output

For each test case, print one line containing the probability that Alice chooses a starting position that leads to the same final card. Your output should have an absolute error of at most 10^{-7}.

Example

input

```
5 2
2 3 5 3 Q
1 1
A
1 2
A
1 10
A
6 1
2 2 2 2 2 2
7 1
2 2 2 2 2 2 2
3 10
10 J K
```

output

```
0.4871377757023325348071573
0.1000000000000000000000000
0.1000000000000000000000000
0.1748923357025314239697490
0.5830713210321767445117468
0.6279229611115749556280350
0.3346565827603272001891974
```

B Jzzhu and Cities

time limit per test: 2 seconds
memory limit per test: 256 megabytes

Jzzhu is the president of country A. There are n cities numbered from 1 to n in his country. City 1 is the capital of A. Also there are m roads connecting the cities. One can go from city u_i to v_i (and vise versa) using the i-th road, the length of this road is x_i. Finally, there are k train routes in the country. One can use the i-th train route to go from capital of the country to city s_i (and vise versa), the length of this route is y_i.

Jzzhu doesn't want to waste the money of the country, so he is going to close some of the train routes. Please tell Jzzhu the maximum number of the train routes which can be closed under the following condition: the length of the shortest path from every city to the capital mustn't change.

Input

The first line contains three integers n, m, k ($2 \le n \le 10^5$; $1 \le m \le 3 \cdot 10^5$; $1 \le k \le 10^5$).

Each of the next m lines contains three integers u_i, v_i, x_i ($1 \le u_i, v_i \le n$; $u_i \ne v_i$; $1 \le x_i \le 10^9$).

Each of the next k lines contains two integers s_i and y_i ($2 \le s_i \le n$; $1 \le y_i \le 10^9$).

It is guaranteed that there is at least one way from every city to the capital. Note, that there can be multiple roads between two cities. Also, there can be multiple routes going to the same city from the capital.

Output

Output a single integer representing the maximum number of the train routes which can be closed.

Examples

input
```
5 5 3
1 2 1
2 3 2
1 3 3
3 4 4
1 5 5
3 5
4 5
5 5
```
output
```
2
```

input
```
2 2 3
1 2 2
2 1 3
2 1
2 2
2 3
```
output
```
2
```

C. New Year Santa Network

2 seconds, 256 megabytes

New Year is coming in Tree World! In this world, as the name implies, there are n cities connected by $n - 1$ roads, and for any two distinct cities there always exists a path between them. The cities are numbered by integers from 1 to n, and the roads are numbered by integers from 1 to $n - 1$. Let's define $d(u, v)$ as total length of roads on the path between city u and city v.

As an annual event, people in Tree World repairs exactly one road per year. As a result, the length of one road decreases. It is already known that in the i-th year, the length of the r_i-th road is going to become w_i, which is shorter than its length before. Assume that the current year is year 1.

Three Santas are planning to give presents annually to all the children in Tree World. In order to do that, they need some preparation, so they are going to choose three distinct cities c_1, c_2, c_3 and make exactly one warehouse in each city. The k-th $(1 \le k \le 3)$ Santa will take charge of the warehouse in city c_k.

It is really boring for the three Santas to keep a warehouse alone. So, they decided to build an only-for-Santa network! The cost needed to build this network equals to $d(c_1, c_2) + d(c_2, c_3) + d(c_3, c_1)$ dollars. Santas are too busy to find the best place, so they decided to choose c_1, c_2, c_3 randomly uniformly over all triples of distinct numbers from 1 to n. Santas would like to know the expected value of the cost needed to build the network.

However, as mentioned, each year, the length of exactly one road decreases. So, the Santas want to calculate the expected after each length change. Help them to calculate the value.

Input

The first line contains an integer n $(3 \le n \le 10^5)$ — the number of cities in Tree World.

Next $n - 1$ lines describe the roads. The i-th line of them $(1 \le i \le n - 1)$ contains three space-separated integers a_i, b_i, l_i $(1 \le a_i, b_i \le n, a_i \ne b_i, 1 \le l_i \le 10^3)$, denoting that the i-th road connects cities a_i and b_i, and the length of i-th road is l_i.

The next line contains an integer q $(1 \le q \le 10^5)$ — the number of road length changes.

Next q lines describe the length changes. The j-th line of them $(1 \le j \le q)$ contains two space-separated integers r_j, w_j $(1 \le r_j \le n - 1, 1 \le w_j \le 10^3)$. It means that in the j-th repair, the length of the r_j-th road becomes w_j. It is guaranteed that w_j is smaller than the current length of the r_j-th road. The same road can be repaired several times.

Output

Output q numbers. For each given change, print a line containing the expected cost needed to build the network in Tree World. The answer will be considered correct if its absolute and relative error doesn't exceed 10^{-6}.

input

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

output

```
14.0000000000
12.0000000000
8.0000000000
6.0000000000
4.0000000000
```

input

```
6
1 5 3
5 3 2
6 1 7
1 4 4
5 2 3
5
1 2
2 1
3 5
4 1
5 2
```

output

```
19.6000000000
18.6000000000
16.6000000000
13.6000000000
12.6000000000
```

Consider the first sample. There are 6 triples: $(1, 2, 3), (1, 3, 2), (2, 1, 3), (2, 3, 1), (3, 1, 2), (3, 2, 1)$. Because $n = 3$, the cost needed to build the network is always $d(1, 2) + d(2, 3) + d(3, 1)$ for all the triples. So, the expected cost equals to $d(1, 2) + d(2, 3) + d(3, 1)$.

Problem D
Divisible Subsequences

Given a sequence of positive integers, count all contiguous subsequences (sometimes called *substrings*, in contrast to *subsequences*, which may leave out elements) the sum of which is divisible by a given number. These subsequences may overlap. For example, the sequence (see sample input)

$$2, 1, 2, 1, 1, 2, 1, 2$$

contains six contiguous subsequences the sum of which is divisible by four: the first to eighth number, the second to fourth number, the second to seventh number, the third to fifth number, the fourth to sixth number, and the fifth to seventh number.

Input

The first line of the input consists of an integer c ($1 \le c \le 200$), the number of test cases. Then follow two lines per test case.
Each test case starts with a line consisting of two integers d ($1 \le d \le 1\,000\,000$) and n ($1 \le n \le 50\,000$), the divisor of the sum of the subsequences and the length of the sequence, respectively. The second line of a test case contains the elements of the sequence, which are integers between 1 and 1 000 000 000, inclusively.

Output

For each test case, print a single line consisting of a single integer, the number of contiguous subsequences the sum of which is divisible by d.

Sample Input

```
2
7 3
1 2 3
4 8
2 1 2 1 1 2 1 2
```

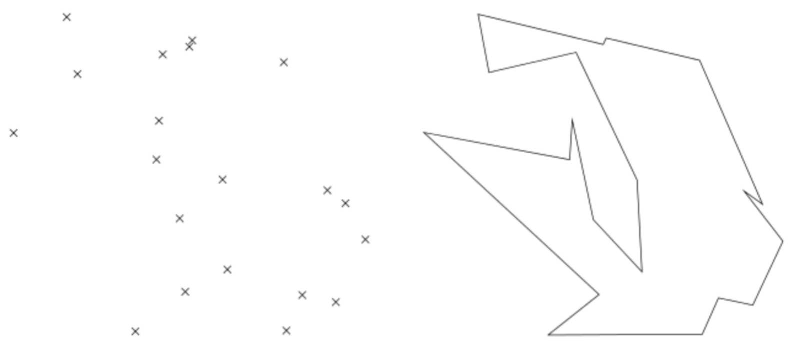
Sample Output

```
0
6
```

Problem E

Simple Polygon

Write a program that constructs a polygon from a given set of points in the plane. Each point of the set has to be a vertex of the polygon, and the polygon must not have any other vertices. No two line segments of the polygon may have any point in common, except for the middle vertex of two consecutive line segments. For example, given the points on the left-hand side, a valid polygon is shown on the right-hand side:



A valid polygon is guaranteed to exist, and any valid polygon will be accepted as a correct answer. Moreover, no two points will share the same location, and not all points will be collinear.

Input

The first line of the input consists of an integer c ($1 \le c \le 200$), the number of test cases. Then follow the test cases, one per line.
Each test case starts with an integer n ($3 \le n \le 2\,000$), the number of given points. Then follow, on the same line, the coordinates of the points. Each point is specified by two integers x and y in the range $-10\,000$ to $10\,000$, inclusively.

Output

For each test case, print a single line containing a permutation of the numbers 0 to $n-1$. Each of these numbers represents the index of a point, in the same order as given in the input. When drawing line segments between consecutive points in the order given by this permutation, the result must be a valid polygon.

Sample Input

```
2
4 0 0 2 0 0 1 1 0
5 0 0 10 0 10 5 5 -1 0 5
```

Sample Output

```
0 3 1 2
3 1 2 4 0
```

F Road to Cinema

time limit per test: 1 second

memory limit per test: 256 megabytes

Vasya is currently at a car rental service, and he wants to reach cinema. The film he has bought a ticket for starts in t minutes. There is a straight road of length s from the service to the cinema. Let's introduce a coordinate system so that the car rental service is at the point 0, and the cinema is at the point s.

There are k gas stations along the road, and at each of them you can fill a car with any amount of fuel for free! Consider that this operation doesn't take any time, i.e. is carried out instantly.

There are n cars in the rental service, i-th of them is characterized with two integers c_i and v_i — the price of this car rent and the capacity of its fuel tank in liters. It's not allowed to fuel a car with more fuel than its tank capacity v_i. All cars are completely fueled at the car rental service.

Each of the cars can be driven in one of two speed modes: normal or accelerated. In the normal mode a car covers 1 kilometer in 2 minutes, and consumes 1 liter of fuel. In the accelerated mode a car covers 1 kilometer in 1 minutes, but consumes 2 liters of fuel. The driving mode can be changed at any moment and any number of times.

Your task is to choose a car with minimum price such that Vasya can reach the cinema before the show starts, i.e. not later than in t minutes. Assume that all cars are completely fueled initially.

Input

The first line contains four positive integers n, k, s and t ($1 \le n \le 2 \cdot 10^5$, $1 \le k \le 2 \cdot 10^5$, $2 \le s \le 10^9$, $1 \le t \le 2 \cdot 10^9$) — the number of cars at the car rental service, the number of gas stations along the road, the length of the road and the time in which the film starts.

Each of the next n lines contains two positive integers c_i and v_i ($1 \le c_i, v_i \le 10^9$) — the price of the i-th car and its fuel tank capacity.

The next line contains k **distinct** integers $g_1, g_2, ..., g_k$ ($1 \le g_i \le s - 1$) — the positions of the gas stations on the road in arbitrary order.

Output

Print the minimum rent price of an appropriate car, i.e. such car that Vasya will be able to reach the cinema before the film starts (not later than in t minutes). If there is no appropriate car, print `-1`.

Examples

input

```
3 1 8 10
10 8
5 7
11 9
3
```

output

```
10
```

input

```
2 2 10 18
10 4
20 6
5 3
```

output

```
20
```

Note

In the first sample, Vasya can reach the cinema in time using the first or the third cars, but it would be cheaper to choose the first one. Its price is equal to 10, and the capacity of its fuel tank is 8. Then Vasya can drive to the first gas station in the accelerated mode in 3 minutes, spending 6 liters of fuel. After that he can full the tank and cover 2 kilometers in the normal mode in 4 minutes, spending 2 liters of fuel. Finally, he drives in the accelerated mode covering the remaining 3 kilometers in 3 minutes and spending 6 liters of fuel.

G. Bear and Different Names

1 second, 256 megabytes

In the army, it isn't easy to form a group of soldiers that will be effective on the battlefield. The communication is crucial and thus no two soldiers should share a name (what would happen if they got an order that Bob is a scouter, if there are two Bobs?).

A group of soldiers is effective if and only if their names are different. For example, a group (John, Bob, Limak) would be effective, while groups (Gary, Bob, Gary) and (Alice, Alice) wouldn't.

You are a spy in the enemy's camp. You noticed n soldiers standing in a row, numbered 1 through n. The general wants to choose a group of k consecutive soldiers. For every k consecutive soldiers, the general wrote down whether they would be an effective group or not.

You managed to steal the general's notes, with $n - k + 1$ strings $s_1, s_2, ..., s_{n-k+1}$, each either "`YES`" or "`NO`".

- The string s_1 describes a group of soldiers 1 through k ("`YES`" if the group is effective, and "`NO`" otherwise).
- The string s_2 describes a group of soldiers 2 through $k + 1$.
- And so on, till the string s_{n-k+1} that describes a group of soldiers $n - k + 1$ through n.

Your task is to find possible names of n soldiers. Names should match the stolen notes. Each name should be a string that consists of between 1 and 10 English letters, inclusive. The first letter should be uppercase, and all other letters should be lowercase. Names don't have to be existing names — it's allowed to print "`Xyzzzdj`" or "`T`" for example.

Find and print any solution. It can be proved that there always exists at least one solution.

Input

The first line of the input contains two integers n and k ($2 \le k \le n \le 50$) — the number of soldiers and the size of a group respectively.

The second line contains $n - k + 1$ strings $s_1, s_2, ..., s_{n-k+1}$. The string s_i is "`YES`" if the group of soldiers i through $i + k - 1$ is effective, and "`NO`" otherwise.

Output

Find any solution satisfying all given conditions. In one line print n space-separated strings, denoting possible names of soldiers in the order. The first letter of each name should be uppercase, while the other letters should be lowercase. Each name should contain English letters only and has length from 1 to 10.

If there are multiple valid solutions, print any of them.

input

```
8 3
NO NO YES YES YES NO
```

output

```
Adam Bob Bob Cpqepqwer Limak Adam Bob Adam
```

input

```
9 8
YES NO
```

output

```
R Q Ccccccccc Ccocc Ccc So Strong Samples Ccc
```

input

```
3 2
NO NO
```

output

```
Na Na Na
```

In the first sample, there are 8 soldiers. For every 3 consecutive ones we know whether they would be an effective group. Let's analyze the provided sample output:

- First three soldiers (i.e. Adam, Bob, Bob) wouldn't be an effective group because there are two Bobs. Indeed, the string s_1 is "`NO`".
- Soldiers 2 through 4 (Bob, Bob, Cpqepqwer) wouldn't be effective either, and the string s_2 is "`NO`".
- Soldiers 3 through 5 (Bob, Cpqepqwer, Limak) would be effective, and the string s_3 is "`YES`".
- ...,
- Soldiers 6 through 8 (Adam, Bob, Adam) wouldn't be effective, and the string s_6 is "`NO`".