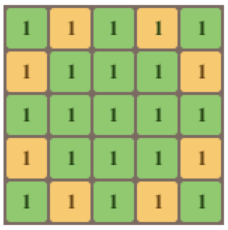
A1. Good Matrix Elements

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

The Smart Beaver from ABBYY got hooked on square matrices. Now he is busy studying an $n \times n$ size matrix, where n is odd. The Smart Beaver considers the following matrix elements good:

- Elements of the main diagonal.
- Elements of the secondary diagonal.
- Elements of the "middle" row — the row which has exactly $\frac{n-1}{2}$ rows above it and the same number of rows below it.
- Elements of the "middle" column — the column that has exactly $\frac{n-1}{2}$ columns to the left of it and the same number of columns to the right of it.



The figure shows a 5×5 matrix. The good elements are marked with green.

Help the Smart Beaver count the sum of good elements of the given matrix.

Input

The first line of input data contains a single odd integer n. Each of the next n lines contains n integers a_{ij} $(0 \le a_{ij} \le 100)$ separated by single spaces — the elements of the given matrix.

The input limitations for getting 30 points are:

- $1 \le n \le 5$

The input limitations for getting 100 points are:

- $1 \le n \le 101$

Output

Print a single integer — the sum of good matrix elements.

Sample test(s)

input

```
3
1 2 3
4 5 6
7 8 9
```

output

```
45
```

input

```
5
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
```

output

```
17
```

Note

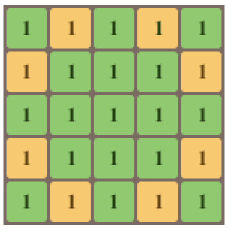
In the first sample all matrix elements will be good. Good elements in the second sample are shown on the figure.

A2. Good Matrix Elements

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

The Smart Beaver from ABBYY got hooked on square matrices. Now he is busy studying an $n \times n$ size matrix, where n is odd. The Smart Beaver considers the following matrix elements good:

- Elements of the main diagonal.
- Elements of the secondary diagonal.
- Elements of the "middle" row — the row which has exactly $\frac{n-1}{2}$ rows above it and the same number of rows below it.
- Elements of the "middle" column — the column that has exactly $\frac{n-1}{2}$ columns to the left of it and the same number of columns to the right of it.



The figure shows a 5×5 matrix. The good elements are marked with green.

Help the Smart Beaver count the sum of good elements of the given matrix.

Input

The first line of input data contains a single odd integer n. Each of the next n lines contains n integers a_{ij} $(0 \le a_{ij} \le 100)$ separated by single spaces — the elements of the given matrix.

The input limitations for getting 30 points are:

- $1 \le n \le 5$

The input limitations for getting 100 points are:

- $1 \le n \le 101$

Output

Print a single integer — the sum of good matrix elements.

Sample test(s)

input

```
3
1 2 3
4 5 6
7 8 9
```

output

```
45
```

input

```
5
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
```

output

```
17
```

Note

In the first sample all matrix elements will be good. Good elements in the second sample are shown on the figure.

B1. Rectangular Game

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

The Smart Beaver from ABBYY decided to have a day off. But doing nothing the whole day turned out to be too boring, and he decided to play a game with pebbles. Initially, the Beaver has n pebbles. He arranges them in a equal rows, each row has b pebbles ($a > 1$). Note that the Beaver must use all the pebbles he has, i. e. $n = a \cdot b$.

10 pebbles are arranged in two rows, each row has 5 pebbles

Once the Smart Beaver has arranged the pebbles, he takes back any of the resulting rows (that is, b pebbles) and discards all other pebbles. Then he arranges **all** his pebbles again (possibly choosing other values of a and b) and takes back one row, and so on. The game continues until at some point the Beaver ends up with exactly one pebble.

The game process can be represented as a finite sequence of integers $c_1, ..., c_k$, where:

- $c_1 = n$
- c_{i+1} is the number of pebbles that the Beaver ends up with after the i-th move, that is, the number of pebbles in a row after some arrangement of c_i pebbles ($1 \le i < k$). Note that $c_i > c_{i+1}$.
- $c_k = 1$

The result of the game is the sum of numbers c_i. You are given n. Find the maximum possible result of the game.

Input

The single line of the input contains a single integer n — the initial number of pebbles the Smart Beaver has.

The input limitations for getting 30 points are:

- $2 \le n \le 50$

The input limitations for getting 100 points are:

- $2 \le n \le 10^9$

Output

Print a single number — the maximum possible result of the game.

Sample test(s)

```
input
10
output
16
```

```
input
8
output
15
```

Note

Consider the first example ($c_1 = 10$). The possible options for the game development are:

- Arrange the pebbles in 10 rows, one pebble per row. Then $c_2 = 1$, and the game ends after the first move with the result of 11.
- Arrange the pebbles in 5 rows, two pebbles per row. Then $c_2 = 2$, and the game continues. During the second move we have two pebbles which can be arranged in a unique way (remember that you are not allowed to put all the pebbles in the same row!) — 2 rows, one pebble per row. $c_3 = 1$, and the game ends with the result of 13.
- Finally, arrange the pebbles in two rows, five pebbles per row. The same logic leads us to $c_2 = 5$, $c_3 = 1$, and the game ends with the result of 16 — the maximum possible result.

B2. Rectangular Game

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

The Smart Beaver from ABBYY decided to have a day off. But doing nothing the whole day turned out to be too boring, and he decided to play a game with pebbles. Initially, the Beaver has n pebbles. He arranges them in a equal rows, each row has b pebbles ($a > 1$). Note that the Beaver must use all the pebbles he has, i. e. $n = a \cdot b$.

10 pebbles are arranged in two rows, each row has 5 pebbles

Once the Smart Beaver has arranged the pebbles, he takes back any of the resulting rows (that is, b pebbles) and discards all other pebbles. Then he arranges **all** his pebbles again (possibly choosing other values of a and b) and takes back one row, and so on. The game continues until at some point the Beaver ends up with exactly one pebble.

The game process can be represented as a finite sequence of integers $c_1, ..., c_k$, where:

- $c_1 = n$
- c_{i+1} is the number of pebbles that the Beaver ends up with after the i-th move, that is, the number of pebbles in a row after some arrangement of c_i pebbles ($1 \le i < k$). Note that $c_i > c_{i+1}$.
- $c_k = 1$

The result of the game is the sum of numbers c_i. You are given n. Find the maximum possible result of the game.

Input
The single line of the input contains a single integer n — the initial number of pebbles the Smart Beaver has.

The input limitations for getting 30 points are:

- $2 \le n \le 50$

The input limitations for getting 100 points are:

- $2 \le n \le 10^9$

Output
Print a single number — the maximum possible result of the game.

Sample test(s)

```
input
10
output
16
```

```
input
8
output
15
```

Note
Consider the first example ($c_1 = 10$). The possible options for the game development are:

- Arrange the pebbles in 10 rows, one pebble per row. Then $c_2 = 1$, and the game ends after the first move with the result of 11.
- Arrange the pebbles in 5 rows, two pebbles per row. Then $c_2 = 2$, and the game continues. During the second move we have two pebbles which can be arranged in a unique way (remember that you are not allowed to put all the pebbles in the same row!) — 2 rows, one pebble per row. $c_3 = 1$, and the game ends with the result of 13.
- Finally, arrange the pebbles in two rows, five pebbles per row. The same logic leads us to $c_2 = 5$, $c_3 = 1$, and the game ends with the result of 16 — the maximum possible result.

C1. Party

time limit per test: 2 seconds

memory limit per test: 256 megabytes

input: standard input

output: standard output

To celebrate the second ABBYY Cup tournament, the Smart Beaver decided to throw a party. The Beaver has a lot of acquaintances, some of them are friends with each other, and some of them dislike each other. To make party successful, the Smart Beaver wants to invite only those of his friends who are connected by friendship relations, and not to invite those who dislike each other. Both friendship and dislike are mutual feelings.

More formally, for each invited person the following conditions should be fulfilled:

- all his friends should also be invited to the party;
- the party shouldn't have any people he dislikes;
- all people who are invited to the party should be connected with him by friendship either directly or through a chain of common friends of arbitrary length. We'll say that people a_1 and a_p are connected through a chain of common friends if there exists a sequence of people $a_2, a_3, ..., a_{p-1}$ such that all pairs of people a_i and a_{i+1} $(1 \le i < p)$ are friends.

Help the Beaver find the maximum number of acquaintances he can invite.

Input

The first line of input contains an integer n — the number of the Beaver's acquaintances.

The second line contains an integer k $(0 \le k \le min(10^5, \frac{n \cdot (n-1)}{2}))$ — the number of pairs of friends. Next k lines contain space-separated pairs of integers u_i, v_i $(1 \le u_i, v_i \le n, u_i \ne v_i)$ — indices of people who form the i-th pair of friends.

The next line contains an integer m $(0 \le m \le min(10^5, \frac{n \cdot (n-1)}{2}))$ — the number of pairs of people who dislike each other. Next m lines describe pairs of people who dislike each other in the same format as the pairs of friends were described.

Each pair of people is mentioned in the input at most once $(0 \le k + m \le \frac{n \cdot (n-1)}{2})$. In particular, two persons cannot be friends and dislike each other at the same time.

The input limitations for getting 30 points are:

- $2 \le n \le 14$

The input limitations for getting 100 points are:

- $2 \le n \le 2000$

Output

Output a single number — the maximum number of people that can be invited to the party. If a group of people that meets all the requirements is impossible to select, output 0.

Sample test(s)

input

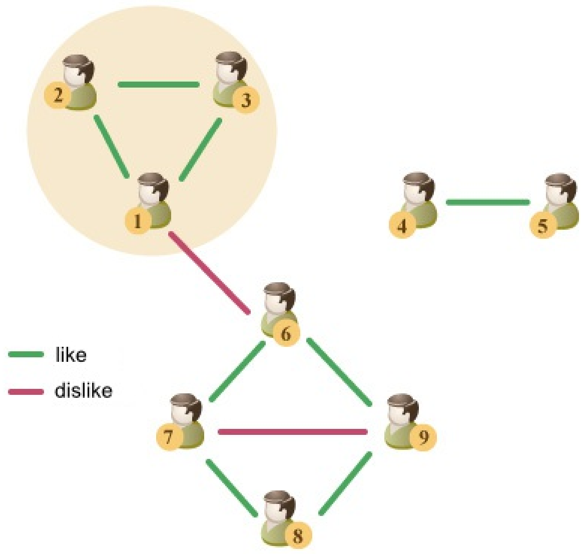
```
9
8
1 2
1 3
2 3
4 5
6 7
7 8
8 9
9 6
2
1 6
7 9
```

output

```
3
```

Note

Let's have a look at the example.



Two groups of people can be invited: $\{1, 2, 3\}$ and $\{4, 5\}$, thus the answer will be the size of the largest of these groups. Group $\{6, 7, 8, 9\}$ doesn't fit, since it includes people 7 and 9 who dislike each other. Group $\{1, 2, 3, 4, 5\}$ also doesn't fit, because not all of its members are connected by a chain of common friends (for example, people 2 and 5 aren't connected).

C2. Party

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

To celebrate the second ABBYY Cup tournament, the Smart Beaver decided to throw a party. The Beaver has a lot of acquaintances, some of them are friends with each other, and some of them dislike each other. To make party successful, the Smart Beaver wants to invite only those of his friends who are connected by friendship relations, and not to invite those who dislike each other. Both friendship and dislike are mutual feelings.

More formally, for each invited person the following conditions should be fulfilled:

- all his friends should also be invited to the party;
- the party shouldn't have any people he dislikes;
- all people who are invited to the party should be connected with him by friendship either directly or through a chain of common friends of arbitrary length. We'll say that people a_1 and a_p are connected through a chain of common friends if there exists a sequence of people $a_2, a_3, ..., a_{p-1}$ such that all pairs of people a_i and a_{i+1} $(1 \le i < p)$ are friends.

Help the Beaver find the maximum number of acquaintances he can invite.

Input

The first line of input contains an integer n — the number of the Beaver's acquaintances.

The second line contains an integer k $(0 \le k \le min(10^5, \frac{n \cdot (n-1)}{2}))$ — the number of pairs of friends. Next k lines contain space-separated pairs of integers u_i, v_i $(1 \le u_i, v_i \le n, u_i \ne v_i)$ — indices of people who form the i-th pair of friends.

The next line contains an integer m $(0 \le m \le min(10^5, \frac{n \cdot (n-1)}{2}))$ — the number of pairs of people who dislike each other. Next m lines describe pairs of people who dislike each other in the same format as the pairs of friends were described.

Each pair of people is mentioned in the input at most once $(0 \le k + m \le \frac{n \cdot (n-1)}{2})$. In particular, two persons cannot be friends and dislike each other at the same time.

The input limitations for getting 30 points are:

- $2 \le n \le 14$

The input limitations for getting 100 points are:

- $2 \le n \le 2000$

Output

Output a single number — the maximum number of people that can be invited to the party. If a group of people that meets all the requirements is impossible to select, output 0.

Sample test(s)

input

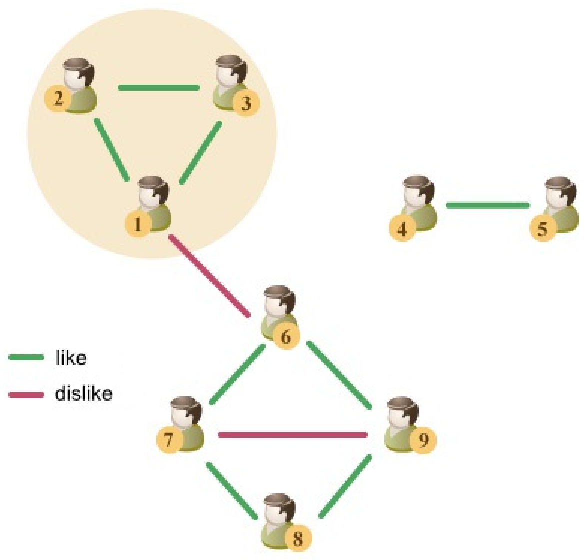
```
9
8
1 2
1 3
2 3
4 5
6 7
7 8
8 9
9 6
2
1 6
7 9
```

output

```
3
```

Note

Let's have a look at the example.



Two groups of people can be invited: $\{1, 2, 3\}$ and $\{4, 5\}$, thus the answer will be the size of the largest of these groups. Group $\{6, 7, 8, 9\}$ doesn't fit, since it includes people 7 and 9 who dislike each other. Group $\{1, 2, 3, 4, 5\}$ also doesn't fit, because not all of its members are connected by a chain of common friends (for example, people 2 and 5 aren't connected).

D1. Encrypting Messages

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

The Smart Beaver from ABBYY invented a new message encryption method and now wants to check its performance. Checking it manually is long and tiresome, so he decided to ask the ABBYY Cup contestants for help.

A message is a sequence of n integers $a_1, a_2, ..., a_n$. Encryption uses a key which is a sequence of m integers $b_1, b_2, ..., b_m$ ($m \le n$). All numbers from the message and from the key belong to the interval from 0 to $c - 1$, inclusive, and all the calculations are performed modulo c.

Encryption is performed in $n - m + 1$ steps. On the first step we add to each number $a_1, a_2, ..., a_m$ a corresponding number $b_1, b_2, ..., b_m$. On the second step we add to each number $a_2, a_3, ..., a_{m+1}$ (changed on the previous step) a corresponding number $b_1, b_2, ..., b_m$. And so on: on step number i we add to each number $a_i, a_{i+1}, ..., a_{i+m-1}$ a corresponding number $b_1, b_2, ..., b_m$. The result of the encryption is the sequence $a_1, a_2, ..., a_n$ after $n - m + 1$ steps.

Help the Beaver to write a program that will encrypt messages in the described manner.

Input

The first input line contains three integers n, m and c, separated by single spaces.

The second input line contains n integers a_i ($0 \le a_i < c$), separated by single spaces — the original message.

The third input line contains m integers b_i ($0 \le b_i < c$), separated by single spaces — the encryption key.

The input limitations for getting 30 points are:

- $1 \le m \le n \le 10^3$
- $1 \le c \le 10^3$

The input limitations for getting 100 points are:

- $1 \le m \le n \le 10^5$
- $1 \le c \le 10^3$

Output

Print n space-separated integers — the result of encrypting the original message.

Sample test(s)

input
```
4 3 2
1 1 1 1
1 1 1
```
output
```
0 1 1 0
```

input
```
3 1 5
1 2 3
4
```
output
```
0 1 2
```

Note

In the first sample the encryption is performed in two steps: after the first step $a = (0, 0, 0, 1)$ (remember that the calculations are performed modulo 2), after the second step $a = (0, 1, 1, 0)$, and that is the answer.

D2. Encrypting Messages

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

The Smart Beaver from ABBYY invented a new message encryption method and now wants to check its performance. Checking it manually is long and tiresome, so he decided to ask the ABBYY Cup contestants for help.

A message is a sequence of n integers $a_1, a_2, ..., a_n$. Encryption uses a key which is a sequence of m integers $b_1, b_2, ..., b_m$ ($m \le n$). All numbers from the message and from the key belong to the interval from 0 to $c - 1$, inclusive, and all the calculations are performed modulo c.

Encryption is performed in $n - m + 1$ steps. On the first step we add to each number $a_1, a_2, ..., a_m$ a corresponding number $b_1, b_2, ..., b_m$. On the second step we add to each number $a_2, a_3, ..., a_{m+1}$ (changed on the previous step) a corresponding number $b_1, b_2, ..., b_m$. And so on: on step number i we add to each number $a_i, a_{i+1}, ..., a_{i+m-1}$ a corresponding number $b_1, b_2, ..., b_m$. The result of the encryption is the sequence $a_1, a_2, ..., a_n$ after $n - m + 1$ steps.

Help the Beaver to write a program that will encrypt messages in the described manner.

Input

The first input line contains three integers n, m and c, separated by single spaces.

The second input line contains n integers a_i ($0 \le a_i < c$), separated by single spaces — the original message.

The third input line contains m integers b_i ($0 \le b_i < c$), separated by single spaces — the encryption key.

The input limitations for getting 30 points are:

- $1 \le m \le n \le 10^3$
- $1 \le c \le 10^3$

The input limitations for getting 100 points are:

- $1 \le m \le n \le 10^5$
- $1 \le c \le 10^3$

Output

Print n space-separated integers — the result of encrypting the original message.

Sample test(s)

input

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

output

```
0 1 1 0
```

input

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

output

```
0 1 2
```

Note

In the first sample the encryption is performed in two steps: after the first step $a = (0, 0, 0, 1)$ (remember that the calculations are performed modulo 2), after the second step $a = (0, 1, 1, 0)$, and that is the answer.

E1. Space Voyage

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

The Smart Beaver from ABBYY plans a space travel on an ultramodern spaceship. During the voyage he plans to visit n planets. For planet i a_i is the maximum number of suitcases that an alien tourist is allowed to bring to the planet, and b_i is the number of citizens on the planet.

The Smart Beaver is going to bring some presents from ABBYY to the planets he will be visiting. The presents are packed in suitcases, x presents in each. The Beaver will take to the ship exactly $a_1 + ... + a_n$ suitcases.

As the Beaver lands on the i-th planet, he takes a_i suitcases and goes out. On the first day on the planet the Beaver takes a walk and gets to know the citizens. On the second and all subsequent days the Beaver gives presents to the citizens — each of the b_i citizens gets one present per day. The Beaver leaves the planet in the evening of the day when the number of presents left is strictly less than the number of citizens (i.e. as soon as he won't be able to give away the proper number of presents the next day). He leaves the remaining presents at the hotel.

The Beaver is going to spend exactly c days traveling. The time spent on flights between the planets is considered to be zero. In how many ways can one choose the positive integer x so that the planned voyage will take exactly c days?

Input

The first input line contains space-separated integers n and c — the number of planets that the Beaver is going to visit and the number of days he is going to spend traveling, correspondingly.

The next n lines contain pairs of space-separated integers a_i, b_i $(1 \le i \le n)$ — the number of suitcases he can bring to the i-th planet and the number of citizens of the i-th planet, correspondingly.

The input limitations for getting 30 points are:

- $1 \le n \le 100$
- $1 \le a_i \le 100$
- $1 \le b_i \le 100$
- $1 \le c \le 100$

The input limitations for getting 100 points are:

- $1 \le n \le 10^4$
- $0 \le a_i \le 10^9$
- $1 \le b_i \le 10^9$
- $1 \le c \le 10^9$

Due to possible overflow, it is recommended to use the 64-bit arithmetic. In some solutions even the 64-bit arithmetic can overflow. So be careful in calculations!

Output

Print a single number k — the number of ways to choose x so as to travel for exactly c days. If there are infinitely many possible values of x, print -1.

Please do not use the `%lld` specifier to read or write 64-bit integers in C++. It is preferred to use `cin`, `cout` streams or the `%I64d` specifier.

Sample test(s)

input

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

output

```
1
```

Note

In the first example there is only one suitable value $x = 5$. Then the Beaver takes 1 suitcase with 5 presents to the first planet. Here he spends 2 days: he hangs around on the first day, and he gives away five presents on the second day. He takes 2 suitcases with 10 presents to the second planet. Here he spends 3 days — he gives away 4 presents on the second and the third days and leaves the remaining 2 presents at the hotel. In total, the Beaver spends 5 days traveling.

For $x = 4$ or less the Beaver won't have enough presents for the second day on the first planet, so the voyage will end too soon. For $x = 6$ and more the Beaver will spend at least one more day on the second planet, and the voyage will take too long.

E2. Space Voyage

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

The Smart Beaver from ABBYY plans a space travel on an ultramodern spaceship. During the voyage he plans to visit n planets. For planet i a_i is the maximum number of suitcases that an alien tourist is allowed to bring to the planet, and b_i is the number of citizens on the planet.

The Smart Beaver is going to bring some presents from ABBYY to the planets he will be visiting. The presents are packed in suitcases, x presents in each. The Beaver will take to the ship exactly $a_1 + ... + a_n$ suitcases.

As the Beaver lands on the i-th planet, he takes a_i suitcases and goes out. On the first day on the planet the Beaver takes a walk and gets to know the citizens. On the second and all subsequent days the Beaver gives presents to the citizens — each of the b_i citizens gets one present per day. The Beaver leaves the planet in the evening of the day when the number of presents left is strictly less than the number of citizens (i.e. as soon as he won't be able to give away the proper number of presents the next day). He leaves the remaining presents at the hotel.

The Beaver is going to spend exactly c days traveling. The time spent on flights between the planets is considered to be zero. In how many ways can one choose the positive integer x so that the planned voyage will take exactly c days?

Input

The first input line contains space-separated integers n and c — the number of planets that the Beaver is going to visit and the number of days he is going to spend traveling, correspondingly.

The next n lines contain pairs of space-separated integers a_i, b_i $(1 \le i \le n)$ — the number of suitcases he can bring to the i-th planet and the number of citizens of the i-th planet, correspondingly.

The input limitations for getting 30 points are:

- $1 \le n \le 100$
- $1 \le a_i \le 100$
- $1 \le b_i \le 100$
- $1 \le c \le 100$

The input limitations for getting 100 points are:

- $1 \le n \le 10^4$
- $0 \le a_i \le 10^9$
- $1 \le b_i \le 10^9$
- $1 \le c \le 10^9$

Due to possible overflow, it is recommended to use the 64-bit arithmetic. In some solutions even the 64-bit arithmetic can overflow. So be careful in calculations!

Output

Print a single number k — the number of ways to choose x so as to travel for exactly c days. If there are infinitely many possible values of x, print -1.

Please do not use the `%lld` specifier to read or write 64-bit integers in C++. It is preferred to use `cin`, `cout` streams or the `%I64d` specifier.

Sample test(s)

input

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

output

```
1
```

Note

In the first example there is only one suitable value $x = 5$. Then the Beaver takes 1 suitcase with 5 presents to the first planet. Here he spends 2 days: he hangs around on the first day, and he gives away five presents on the second day. He takes 2 suitcases with 10 presents to the second planet. Here he spends 3 days — he gives away 4 presents on the second and the third days and leaves the remaining 2 presents at the hotel. In total, the Beaver spends 5 days traveling.

For $x = 4$ or less the Beaver won't have enough presents for the second day on the first planet, so the voyage will end too soon. For $x = 6$ and more the Beaver will spend at least one more day on the second planet, and the voyage will take too long.

F1. Script Generation

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

The Smart Beaver from ABBYY was offered a job of a screenwriter for the ongoing TV series. In particular, he needs to automate the hard decision: which main characters will get married by the end of the series.

There are n single men and n single women among the main characters. An opinion poll showed that viewers like several couples, and a marriage of any of them will make the audience happy. The Smart Beaver formalized this fact as k triples of numbers (h, w, r), where h is the index of the man, w is the index of the woman, and r is the measure of the audience's delight in case of the marriage of this couple. The same poll showed that the marriage of any other couple will leave the audience indifferent, so the screenwriters decided not to include any such marriages in the plot.

The script allows you to arrange several marriages between the heroes or not to arrange marriages at all. A subset of some of the k marriages is considered acceptable if each man and each woman is involved in at most one marriage of the subset (the series won't allow any divorces). The value of the acceptable set of marriages is the total delight the spectators will get from the marriages included in this set.

Obviously, there is a finite number of acceptable sets, and they all describe some variants of the script. The screenwriters do not want to choose a set with maximum value — it would make the plot too predictable. So the Smart Beaver offers the following option: sort all the acceptable sets in increasing order of value and choose the t-th set from the sorted list. Thus, $t = 1$ corresponds to a plot without marriages, $t = 2$ — to a single marriage resulting in minimal delight for the audience, and so on.

Help the Beaver to implement the algorithm for selecting the desired set.

Input

The first input line contains integers n, k and t $(1 \le k \le min(100, n^2)$, $1 \le t \le 2 \cdot 10^5)$, separated by single spaces. Next k lines contain triples of integers (h, w, r) $(1 \le h, w \le n;\ 1 \le r \le 1000)$, separated by single spaces, which describe the possible marriages. It is guaranteed that the input data is correct: t doesn't exceed the total number of acceptable sets, and each pair (h, w) is present in at most one triple.

The input limitations for getting 30 points are:

- $1 \le n \le 5$

The input limitations for getting 100 points are:

- $1 \le n \le 20$

Output

Print a single number — the value of the t-th acceptable variant.

Sample test(s)

input

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

output

```
2
```

input

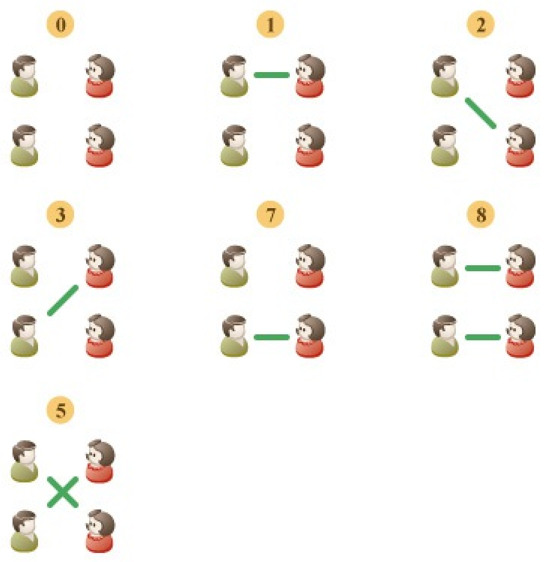
```
2 4 7
1 1 1
1 2 2
2 1 3
2 2 7
```

output

```
8
```

Note

The figure shows 7 acceptable sets of marriages that exist in the first sample.


0
1
2
3
7
8
5

F2. Script Generation

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

The Smart Beaver from ABBYY was offered a job of a screenwriter for the ongoing TV series. In particular, he needs to automate the hard decision: which main characters will get married by the end of the series.

There are n single men and n single women among the main characters. An opinion poll showed that viewers like several couples, and a marriage of any of them will make the audience happy. The Smart Beaver formalized this fact as k triples of numbers (h, w, r), where h is the index of the man, w is the index of the woman, and r is the measure of the audience's delight in case of the marriage of this couple. The same poll showed that the marriage of any other couple will leave the audience indifferent, so the screenwriters decided not to include any such marriages in the plot.

The script allows you to arrange several marriages between the heroes or not to arrange marriages at all. A subset of some of the k marriages is considered acceptable if each man and each woman is involved in at most one marriage of the subset (the series won't allow any divorces). The value of the acceptable set of marriages is the total delight the spectators will get from the marriages included in this set.

Obviously, there is a finite number of acceptable sets, and they all describe some variants of the script. The screenwriters do not want to choose a set with maximum value — it would make the plot too predictable. So the Smart Beaver offers the following option: sort all the acceptable sets in increasing order of value and choose the t-th set from the sorted list. Thus, $t = 1$ corresponds to a plot without marriages, $t = 2$ — to a single marriage resulting in minimal delight for the audience, and so on.

Help the Beaver to implement the algorithm for selecting the desired set.

Input

The first input line contains integers n, k and t $(1 \le k \le min(100, n^2)$, $1 \le t \le 2 \cdot 10^5)$, separated by single spaces. Next k lines contain triples of integers (h, w, r) $(1 \le h, w \le n; 1 \le r \le 1000)$, separated by single spaces, which describe the possible marriages. It is guaranteed that the input data is correct: t doesn't exceed the total number of acceptable sets, and each pair (h, w) is present in at most one triple.

The input limitations for getting 30 points are:

- $1 \le n \le 5$

The input limitations for getting 100 points are:

- $1 \le n \le 20$

Output

Print a single number — the value of the t-th acceptable variant.

Sample test(s)

input

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

output

```
2
```

input

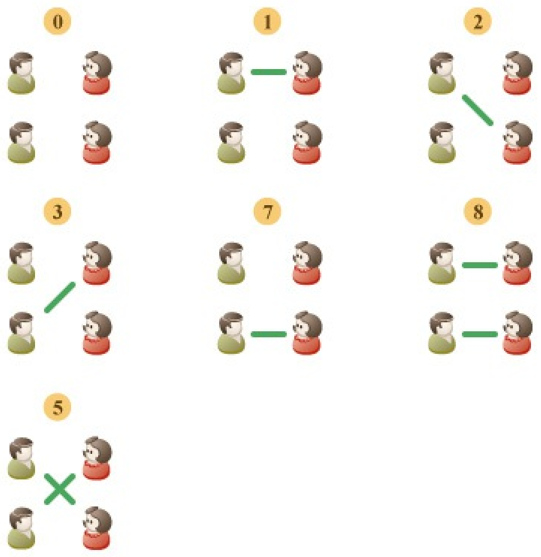
```
2 4 7
1 1 1
1 2 2
2 1 3
2 2 7
```

output

```
8
```

Note

The figure shows 7 acceptable sets of marriages that exist in the first sample.


0
1
2
3
7
8
5

G1. Fibonacci Strings

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Fibonacci strings are defined as follows:

- f_1 = «a»
- f_2 = «b»
- $f_n = f_{n-1} f_{n-2}$, $n > 2$

Thus, the first five Fibonacci strings are: "a", "b", "ba", "bab", "babba".

You are given a Fibonacci string and m strings s_i. For each string s_i, find the number of times it occurs in the given Fibonacci string as a substring.

Input

The first line contains two space-separated integers k and m — the number of a Fibonacci string and the number of queries, correspondingly.

Next m lines contain strings s_i that correspond to the queries. It is guaranteed that strings s_i aren't empty and consist only of characters "a" and "b".

The input limitations for getting 30 points are:

- $1 \le k \le 3000$
- $1 \le m \le 3000$
- The total length of strings s_i doesn't exceed 3000

The input limitations for getting 100 points are:

- $1 \le k \le 10^{18}$
- $1 \le m \le 10^4$
- The total length of strings s_i doesn't exceed 10^5

Please do not use the %lld specifier to read or write 64-bit integers in C++. It is preferred to use cin, cout streams or the %I64d specifier.

Output

For each string s_i print the number of times it occurs in the given Fibonacci string as a substring. Since the numbers can be large enough, print them modulo 1000000007 $(10^9 + 7)$. Print the answers for the strings in the order in which they are given in the input.

Sample test(s)

input

```
6 5
a
b
ab
ba
aba
```

output

```
3
5
3
3
1
```

G2. Fibonacci Strings

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Fibonacci strings are defined as follows:

- f_1 = «a»
- f_2 = «b»
- $f_n = f_{n-1} f_{n-2}$, $n > 2$

Thus, the first five Fibonacci strings are: "a", "b", "ba", "bab", "babba".

You are given a Fibonacci string and m strings s_i. For each string s_i, find the number of times it occurs in the given Fibonacci string as a substring.

Input

The first line contains two space-separated integers k and m — the number of a Fibonacci string and the number of queries, correspondingly.

Next m lines contain strings s_i that correspond to the queries. It is guaranteed that strings s_i aren't empty and consist only of characters "a" and "b".

The input limitations for getting 30 points are:

- $1 \le k \le 3000$
- $1 \le m \le 3000$
- The total length of strings s_i doesn't exceed 3000

The input limitations for getting 100 points are:

- $1 \le k \le 10^{18}$
- $1 \le m \le 10^4$
- The total length of strings s_i doesn't exceed 10^5

Please do not use the %lld specifier to read or write 64-bit integers in C++. It is preferred to use cin, cout streams or the %I64d specifier.

Output

For each string s_i print the number of times it occurs in the given Fibonacci string as a substring. Since the numbers can be large enough, print them modulo 1000000007 $(10^9 + 7)$. Print the answers for the strings in the order in which they are given in the input.

Sample test(s)

input

```
6 5
a
b
ab
ba
aba
```

output

```
3
5
3
3
1
```