A. Watching a movie

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

You have decided to watch the best moments of some movie. There are two buttons on your player:

1. Watch the current minute of the movie. By pressing this button, you watch the current minute of the movie and the player automatically proceeds to the next minute of the movie.
2. Skip exactly x minutes of the movie (x is some fixed positive integer). If the player is now at the t-th minute of the movie, then as a result of pressing this button, it proceeds to the minute $(t + x)$.

Initially the movie is turned on in the player on the first minute, and you want to watch exactly n best moments of the movie, the i-th best moment starts at the l_i-th minute and ends at the r_i-th minute (more formally, the i-th best moment consists of minutes: $l_i, l_i + 1, ..., r_i$).

Determine, what is the minimum number of minutes of the movie you have to watch if you want to watch all the best moments?

Input

The first line contains two space-separated integers n, x ($1 \le n \le 50$, $1 \le x \le 10^5$) — the number of the best moments of the movie and the value of x for the second button.

The following n lines contain the descriptions of the best moments of the movie, the i-th line of the description contains two integers separated by a space l_i, r_i ($1 \le l_i \le r_i \le 10^5$).

It is guaranteed that for all integers i from 2 to n the following condition holds: $r_{i-1} < l_i$.

Output

Output a single number — the answer to the problem.

Sample test(s)

input

```
2 3
5 6
10 12
```

output

```
6
```

input

```
1 1
1 100000
```

output

```
100000
```

Note

In the first sample, the player was initially standing on the first minute. As the minutes from the 1-st to the 4-th one don't contain interesting moments, we press the second button. Now we can not press the second button and skip 3 more minutes, because some of them contain interesting moments. Therefore, we watch the movie from the 4-th to the 6-th minute, after that the current time is 7. Similarly, we again skip 3 minutes and then watch from the 10-th to the 12-th minute of the movie. In total, we watch 6 minutes of the movie.

In the second sample, the movie is very interesting, so you'll have to watch all 100000 minutes of the movie.

B. Lecture

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

You have a new professor of graph theory and he speaks very quickly. You come up with the following plan to keep up with his lecture and make notes.

You know two languages, and the professor is giving the lecture in the first one. The words in both languages consist of lowercase English characters, each language consists of several words. For each language, all words are distinct, i.e. they are spelled differently. Moreover, the words of these languages have a one-to-one correspondence, that is, for each word in each language, there exists exactly one word in the other language having has the same meaning.

You can write down every word the professor says in either the first language or the second language. Of course, during the lecture you write down each word in the language in which the word is shorter. In case of equal lengths of the corresponding words you prefer the word of the first language.

You are given the text of the lecture the professor is going to read. Find out how the lecture will be recorded in your notes.

Input

The first line contains two integers, n and m $(1 \le n \le 3000, 1 \le m \le 3000)$ — the number of words in the professor's lecture and the number of words in each of these languages.

The following m lines contain the words. The i-th line contains two strings a_i, b_i meaning that the word a_i belongs to the first language, the word b_i belongs to the second language, and these two words have the same meaning. It is guaranteed that no word occurs in both languages, and each word occurs in its language exactly once.

The next line contains n space-separated strings $c_1, c_2, ..., c_n$ — the text of the lecture. It is guaranteed that each of the strings c_i belongs to the set of strings $\{a_1, a_2, ... a_m\}$.

All the strings in the input are non-empty, each consisting of no more than 10 lowercase English letters.

Output

Output exactly n words: how you will record the lecture in your notebook. Output the words of the lecture in the same order as in the input.

Sample test(s)

input

```
4 3
codeforces codesecrof
contest round
letter message
codeforces contest letter contest
```

output

```
codeforces round letter round
```

input

```
5 3
joll wuqrd
euzf un
hbnyiyc rsoqqveh
hbnyiyc joll joll euzf joll
```

output

```
hbnyiyc joll joll un joll
```

C. Crazy Town

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

Crazy Town is a plane on which there are n infinite line roads. Each road is defined by the equation $a_ix + b_iy + c_i = 0$, where a_i and b_i are not both equal to the zero. The roads divide the plane into connected regions, possibly of infinite space. Let's call each such region a block. We define an intersection as the point where at least two different roads intersect.

Your home is located in one of the blocks. Today you need to get to the University, also located in some block. In one step you can move from one block to another, if the length of their common border is nonzero (in particular, this means that if the blocks are adjacent to one intersection, but have no shared nonzero boundary segment, then it are not allowed to move from one to another one in one step).

Determine what is the minimum number of steps you have to perform to get to the block containing the university. It is guaranteed that neither your home nor the university is located on the road.

Input

The first line contains two space-separated integers x_1, y_1 ($-10^6 \le x_1, y_1 \le 10^6$) — the coordinates of your home.

The second line contains two integers separated by a space x_2, y_2 ($-10^6 \le x_2, y_2 \le 10^6$) — the coordinates of the university you are studying at.

The third line contains an integer n ($1 \le n \le 300$) — the number of roads in the city. The following n lines contain 3 space-separated integers ($-10^6 \le a_i, b_i, c_i \le 10^6$; $|a_i| + |b_i| > 0$) — the coefficients of the line $a_ix + b_iy + c_i = 0$, defining the i-th road. It is guaranteed that no two roads are the same. In addition, neither your home nor the university lie on the road (i.e. they do not belong to any one of the lines).

Output

Output the answer to the problem.

Sample test(s)

input

```
1 1
-1 -1
2
0 1 0
1 0 0
```

output

```
2
```

input

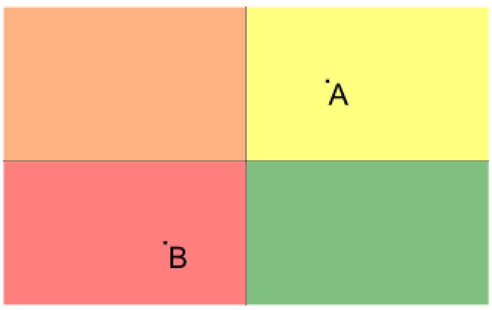
```
1 1
-1 -1
3
1 0 0
0 1 0
1 1 -3
```

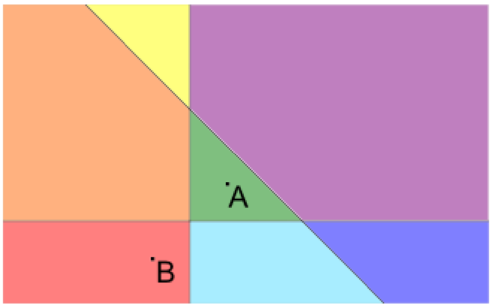
output

```
2
```

Note

Pictures to the samples are presented below (A is the point representing the house; B is the point representing the university, different blocks are filled with different colors):





A
B

D. Name That Tune

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

It turns out that you are a great fan of rock band AC/PE. Peter learned that and started the following game: he plays the first song of the list of n songs of the group, and you have to find out the name of the song. After you tell the song name, Peter immediately plays the following song in order, and so on.

The i-th song of AC/PE has its recognizability p_i. This means that if the song has not yet been recognized by you, you listen to it for exactly one more second and with probability of p_i percent you recognize it and tell it's name. Otherwise you continue listening it. Note that you can only try to guess it only when it is integer number of seconds after the moment the song starts playing.

In all AC/PE songs the first words of chorus are the same as the title, so when you've heard the first t_i seconds of i-th song and its chorus starts, you immediately guess its name for sure.

For example, in the song Highway To Red the chorus sounds pretty late, but the song has high recognizability. In the song Back In Blue, on the other hand, the words from the title sound close to the beginning of the song, but it's hard to name it before hearing those words. You can name both of these songs during a few more first seconds.

Determine the expected number songs of you will recognize if the game lasts for exactly T seconds (i. e. you can make the last guess on the second T, after that the game stops).

If all songs are recognized faster than in T seconds, the game stops after the last song is recognized.

Input

The first line of the input contains numbers n and T ($1 \le n \le 5000$, $1 \le T \le 5000$), separated by a space. Next n lines contain pairs of numbers p_i and t_i ($0 \le p_i \le 100$, $1 \le t_i \le T$). The songs are given in the same order as in Petya's list.

Output

Output a single number — the expected number of the number of songs you will recognize in T seconds. Your answer will be considered correct if its absolute or relative error does not exceed 10^{-6}.

Sample test(s)

input

```
2 2
50 2
10 1
```

output

```
1.500000000
```

input

```
2 2
0 2
100 2
```

output

```
1.000000000
```

input

```
3 3
50 3
50 2
25 2
```

output

```
1.687500000
```

input

```
2 2
0 2
0 2
```

output

```
1.000000000
```

E. Array and Operations

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

You have written on a piece of paper an array of n positive integers $a[1], a[2], ..., a[n]$ and m *good* pairs of integers $(i_1, j_1), (i_2, j_2), ..., (i_m, j_m)$. Each *good* pair (i_k, j_k) meets the following conditions: $i_k + j_k$ is an odd number and $1 \le i_k < j_k \le n$.

In one operation you can perform a sequence of actions:

- take one of the *good* pairs (i_k, j_k) and some integer v $(v > 1)$, which divides both numbers $a[i_k]$ and $a[j_k]$;
- divide both numbers by v, i. e. perform the assignments: $a[i_k] = \frac{a[i_k]}{v}$ and $a[j_k] = \frac{a[j_k]}{v}$.

Determine the maximum number of operations you can sequentially perform on the given array. Note that one pair may be used several times in the described operations.

Input

The first line contains two space-separated integers n, m $(2 \le n \le 100, 1 \le m \le 100)$.

The second line contains n space-separated integers $a[1], a[2], ..., a[n]$ $(1 \le a[i] \le 10^9)$ — the description of the array.

The following m lines contain the description of *good* pairs. The k-th line contains two space-separated integers i_k, j_k $(1 \le i_k < j_k \le n, i_k + j_k$ is an odd number).

It is guaranteed that all the *good* pairs are distinct.

Output

Output the answer for the problem.

Sample test(s)

input

```
3 2
8 3 8
1 2
2 3
```

output

```
0
```

input

```
3 2
8 12 8
1 2
2 3
```

output

```
2
```