**Codeforces Round #284 (Div. 2)**

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 *li*-th minute and ends at the *ri*-th minute (more formally, the *i*-th best moment consists of minutes: *li*, *li* + 1, ..., *ri*).

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**

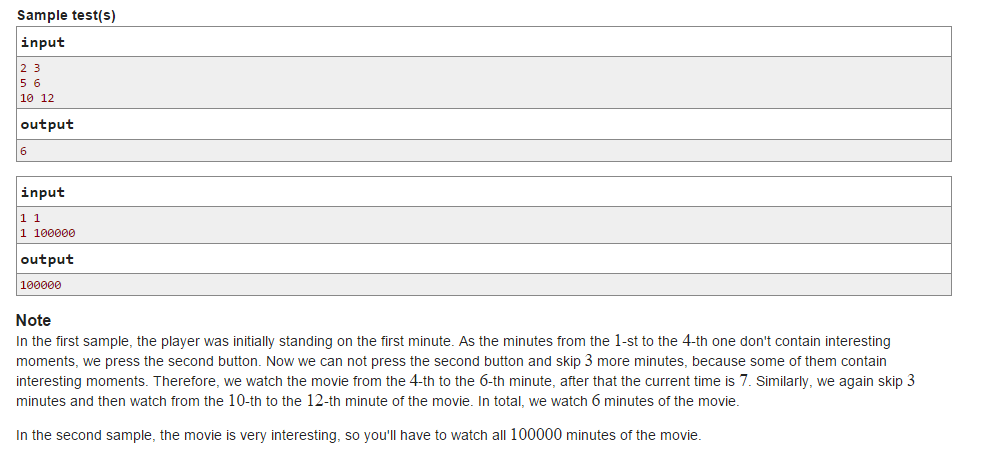
The first line contains two space-separated integers *n*, *x* (1 ≤ *n* ≤ 50, 1 ≤ *x* ≤ 105) — the number of the best moments of the movie and the value of *x* for the second button.

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 *li*, *ri* (1 ≤ *li* ≤ *ri* ≤ 105).

It is guaranteed that for all integers *i* from 2 to *n* the following condition holds: *ri*- 1 < *li*.

**Output**

Output a single number — the answer to the problem.



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 ≤ *n* ≤ 3000, 1 ≤ *m* ≤ 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 *ai*, *bi* meaning that the word *ai* belongs to the first language, the word *bi*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, ..., *cn* — the text of the lecture. It is guaranteed that each of the strings *ci* belongs to the set of strings {*a*1, *a*2, ... *am*}.

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

**Output**

Output exactly *n* rows — the description of how you will record the lecture in your notebook. Output the words of the lecture in the same order as in the input.



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 *aix* + *biy* + *ci* = 0, where *ai* and *bi* 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**

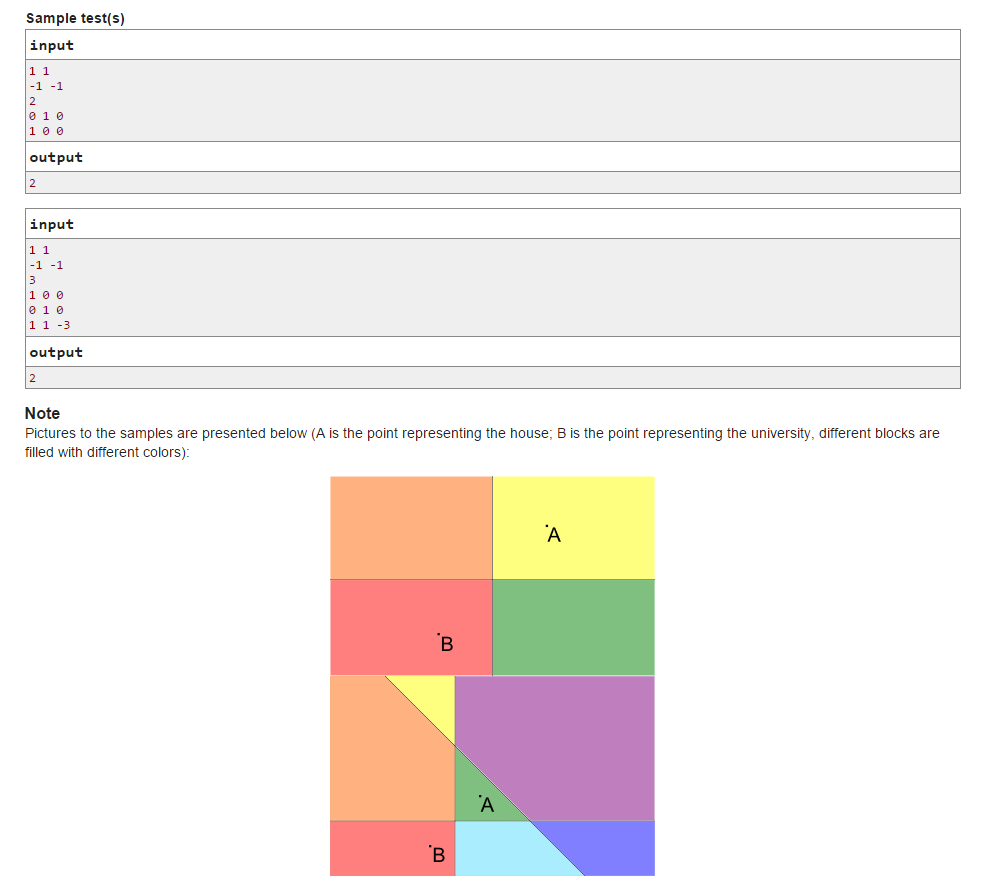
The first line contains two space-separated integers *x*1, *y*1 ( - 106 ≤ *x*1, *y*1 ≤ 106) — the coordinates of your home.

The second line contains two integers separated by a space *x*2, *y*2 ( - 106 ≤ *x*2, *y*2 ≤ 106) — the coordinates of the university you are studying at.

The third line contains an integer *n* (1 ≤ *n* ≤ 300) — the number of roads in the city. The following *n* lines contain 3 space-separated integers ( - 106 ≤ *ai*, *bi*, *ci* ≤ 106; |*ai*| + |*bi*| > 0) — the coefficients of the line *aix* + *biy* + *ci* = 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.



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 *pi*. 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 *pi* 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 *ti* 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).

**Input**

The first line of the input contains numbers *n* and *T* (1 ≤ *n* ≤ 5000, 1 ≤ *T* ≤ 5000), separated by a space. Next *n* lines contain pairs of numbers *pi*and *ti* (0 ≤ *pi* ≤ 100, 1 ≤ *ti* ≤ *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.

