Problem A. Clothes

Input file:	`clothes.in`
Output file:	`clothes.out`
Time limit:	3 seconds
Memory limit:	256 megabytes

Andrew always checks the weather forecast before going somewhere. He uses his favorite website which always gives him correct forecasts. It's unbelievable, but the website never makes mistakes! Unfortunately it doesn't say exact temperature, but gives only the lowest possible temperature for a day. We will use a simplified model where the temperature throughout the day is constant. In other words, for each day, Andrew knows such value t that the temperature throughout that day will be equal to some $t' \geqslant t$, but he will only find out the value of t' when the day comes.

You don't need any clothes when it's more than $+30\,^{\circ}$C outside, so you may assume that the highest possible temperature always equals $+30\,^{\circ}$C. But even knowing that the lowest temperature is not very low can help to bring less warm garments (the heaviest things he usually takes).

There are several types of garments Andrew can wear. For each type of garment Andrew knows the lowest temperature it will still protect him at, and the highest temperature he can still tolerate wearing it at. Let's suppose he can't put one warm garment on top of another (it's not always convenient to wear two jackets). Andrew doesn't want to wash his clothes in the trip, so he wants to wear every garment for at most *one day*, but of course he may bring several garments of the same type.

What is the minimum total number of garments he needs to bring to be prepared for every combination of actual weather for each day?

Input

On the first line you are given the number of days n ($1 \leqslant n \leqslant 10^5$) Andrew wants to spend in his trip. The next line contains n floating-point numbers t_i ($-30 \leqslant t_i < 30$) — the lowest possible temperature for each day. On the third line you are given the number of available garment types m ($1 \leqslant m \leqslant 10^5$). Each of the next m lines contains two floating-point numbers l_i, r_i ($-30 \leqslant l_i < r_i \leqslant 30$) — the bounds of the temperature interval for each garment type. All floating-point numbers have at most 6 digits after the decimal point.

Output

Print the minimum number of garments required to be guaranteed to survive the entire trip. In case it's impossible, output -1.

Examples

clothes.in	clothes.out

2
-10.5 -5.3
2
-8 30
-12 0
	3

2
-10.5 -5.3
1
10 30
	-1

Note

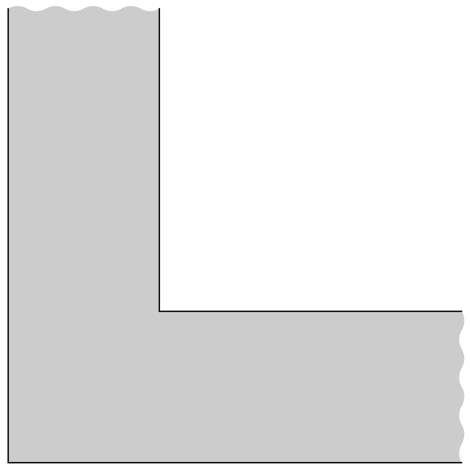
In the first example, Andrew should bring two garments of “from -8 to 30” type and one garment of “from -12 to 0” type.

Problem B. Crosshair

Input file:	`crosshair.in`
Output file:	`crosshair.out`
Time limit:	6 seconds
Memory limit:	256 megabytes

While bored on a maths class, Andrew has drawn n points in his notebook. He is now wondering: is it possible to cover all those points using his corner ruler, possibly rotating it beforehand?

The ruler he has has the following form:



It is infinite both to the right and to the top. If we introduce a coordinate system that has origin in the bottom-left corner of the ruler, and axes along its sides, then the ruler consists of points such that $x \geqslant 0$, $y \geqslant 0$ and either $x \leqslant r$, or $y \leqslant r$, or both. This number r is called the size of the ruler.

A point is said to be covered by the ruler if it lies inside it or on its border.

Input

The first line of the input file contains two integers n ($1 \leqslant n \leqslant 1\,000$) and r ($1 \leqslant r \leqslant 1\,000$), denoting the number of points and the size of the ruler. The next n lines contain two integers each — the coordinates of the points. All coordinates don't exceed $1\,000$ by absolute value.

Output

If it is possible to cover all points, output '`FIRE`' on the first line of output file, otherwise output '`BORING`'.

When a solution exists, output the coordinates of the outer corner of the ruler on the second line, and the coordinates of the inner corner of the ruler on the third line. Please output all numbers as precisely as possible. The verifier will check if each point is inside the ruler, or is at most 10^{-6} away from it. It is guaranteed that when it's impossible to cover all points by the ruler, for every position of the ruler there will be at least one point that is at least 10^{-5} away from it.

In case there are several optimal solutions, output any.

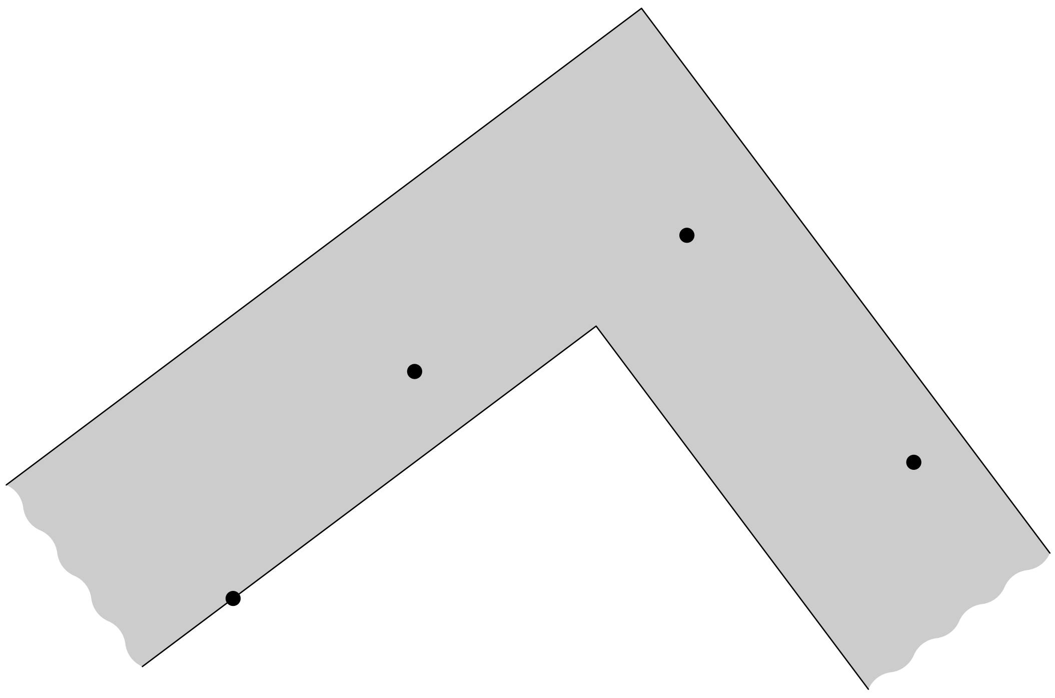
Examples

crosshair.in	crosshair.out

4 5
0 2
4 7
15 5
10 10
	FIRE
	9.0 15.0
	8.0 8.0

Note

Here's what happens in the example case:



Problem C. Floyd

Input file:	`floyd.in`
Output file:	`floyd.out`
Time limit:	6 seconds
Memory limit:	256 megabytes

Andrew is studying random graphs, and now he wants to find shortest paths in such graphs. More specifically, he generates a random directed graph, and wants to find the length of the shortest path between all pairs of vertices.

The graph is a complete directed graph generated in the following manner. First, Andrew chooses the number of vertices n and a random seed s. Then, the following algorithm is used to find the arc lengths:

```
for i from 1 to n {
  for j from 1 to n {
    if i != j then {
      d[i][j] = s % 1000
      s = (s * 16807) % 2147483647
    }
  }
}
```

Sufficiently large integer types are used for all computations, and `d[i][j]` is the length of the arc from vertex i to vertex j. Notice that all lengths generated in this way are between 0 and 999.

Now Andrew wants you to find the lengths of the shortest paths between all pairs of vertices. Since there might be many such pairs, you will need to output one number instead of n^2 numbers. More specifically, let `p[i][j]` be the length of the shortest path from vertex i to vertex j (it is equal to 0 when $i = j$). Use the following algorithm to compute the value to output:

```
r = 0
for i from 1 to n {
  for j from 1 to n {
    r = (r * 16807 + p[i][j]) % 2147483647
  }
}
```

You need to find the final value of r.

Input

The first line of the input file contains one integer n, $1 \leqslant n \leqslant 2\,000$. The second line of the input file contains one integer s, $1 \leqslant s \leqslant 2\,147\,483\,646$. The values of s to use in testcases will be chosen randomly (except the one in the sample input).

Output

Output one integer — the final value of r in the above algorithm.

Examples

floyd.in	floyd.out
3 57	532872718

Problem D. FourSquares

Input file:	`four-squares.in`
Output file:	`four-squares.out`
Time limit:	4 seconds
Memory limit:	256 megabytes

Recently geosocial services became very popular in the world. Numberland is no exception from this modern trend. People in the Numberland live in a discrete 4-dimensional world, which consists of finite number of points with coordinates (x, y, z, t), where $0 \leqslant x, y, z, t < p$ for some prime number p. Each resident of the Numberland lives in some such point, and each of the points is inhabited by exactly one Numberlander. The center of Numberland is point $(0, 0, 0, 0)$.

The society in the Numberland has p castes. The overlord of Numberland lives in the center, $(0, 0, 0, 0)$. Then go supreme lords, lords, minor lords and all the other castes. The principle is, the closer you live to the center, the higher is your social status. So, all the people who live at the same distance from the center (that is, all the people for which $x^2 + y^2 + z^2 + t^2 = a \ (mod\ p)$ for some fixed a) belong to the same caste called caste a. Note that the distance is the square of the Euclidean distance, but with a nuance: the distance is taken modulo p.

Now the lord of Numberland wants to know how many people are there in a particular caste, because if it is too large, it can become revolutionist, set up a meeting at FourSquares using another popular service, EvilNumberBook, and destroy the order in the Numberland. Please help him with this complex task.

Input

The only line of the input file contains two integers: p and a ($2 \leqslant p \leqslant 10^7$, $0 \leqslant a < p$, p is prime).

Output

Output the amount of people in the caste a in Numberland with p castes modulo 1000000007 ($10^9 + 7$).

Examples

four-squares.in	four-squares.out
2 0	8
2 1	8
5 0	145

Problem E. Unique identification

Input file:	`identification.in`
Output file:	`identification.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Andrew has been developing a new generation of IDs for the government of Antarctica. Each ID is a connected set of unit cubes that is 1 unit cube thick (we will call them *cells*) (a set is *connected* if any cell of the set is reachable from any other cell of the set by going from a cell to its neighbor without leaving the set; neighbors of a cell are 4 cells that share a border with it).

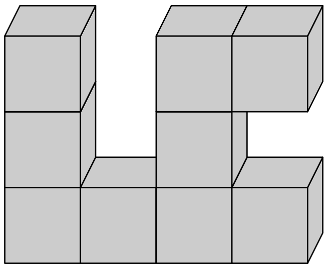


Figure A

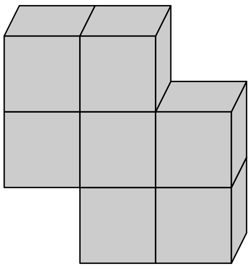
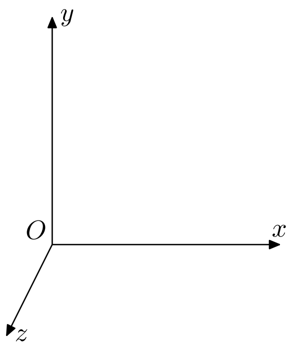


Figure B



Unfortunately, Andrew has noticed that some IDs have the following drawback: one can rotate and/or move them in space in such a way that it's impossible to tell if they have been rotated. For example, after rotating figure B above by 180 degrees around a line parallel to the Oz axis, we get an identical figure so it would be impossible to differentiate the rotated and non-rotated IDs. And since the cubes themselves will contain very sophisticated equipment inside, they will be useless if rotated incorrectly.

Andrew has decided to use colored cubes to assemble each ID in such a way that rotating and/or moving the ID always results in a colored figure different from the original. For example, here are 3 ways of assembling figure B out of cubes of two different colors:

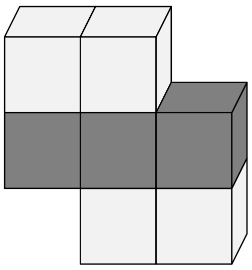


Figure B1

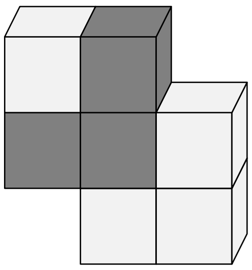


Figure B2

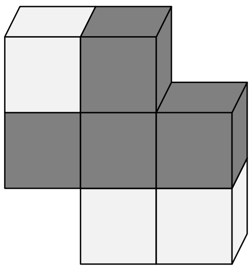


Figure B3

Figure B1 is still a bad ID, since it can be rotated by 180 degrees around a line parallel to Oz axis and still appear the same. Figure B2 is also a bad ID, since it can be rotated (using the third dimension) along a line parallel to the top-left to bottom-right diagonal in the Oxy plane and still appear the same. Figure B3 is finally a good ID since no rotation will result in the same figure.

What is the minimum number of different colors required to make the given figure a good ID?

Input

The first line of the input file contains two integers h and w ($1 \leqslant h, w \leqslant 8$) denoting the height and width of the figure. The next h lines contain w characters each, describing the figure. Each character is either

'*', denoting a cell that is a part of the figure, or '.', denoting an empty cell. There will be at least one '*' character in every row and in every column, and all '*' characters will form a connected figure.

Output

On the first line of the output file write the minimum number of colors required, or -1 if it's impossible to make a good ID out of this figure.

If a solution exists, on the next h lines output the colored figure in the same way it is written in the input file, but replacing '*' characters with capital English letters ('A' through 'Z') with different letters denoting different colors. It is guaranteed that when a solution exists, 26 colors is enough.

In case there are several possible optimal solutions, output any.

Examples

identification.in	identification.out
3 4 *.** *.*. ****	1 Q.QQ Q.Q. QQQQ
3 3 **. *** .**	2 AB. BBB .AA
1 1 *	-1

Problem F. Mega Nim

Input file:	`meganim.in`
Output file:	`meganim.out`
Time limit:	8 seconds
Memory limit:	256 megabytes

Andrew was amazed by the classification of losing positions in the famous game of Nim. The rules of the game are not important in this problem; the only thing you need to know about it that its positions are ordered n-tuples of non-negative integers $a_1, a_2, \ldots, a_n$, and the amazing fact Andrew has just learned: losing positions in the game of Nim are those where $a_1 \oplus a_2 \oplus \cdots \oplus a_n = 0$ (where $\oplus$ stands for bitwise exclusive or).

Please help Andrew to find the number of losing positions with all $0 \leqslant a_i \leqslant M$. Since this number can be very big, please find it modulo $1\,000\,000\,007 = 10^9 + 7$.

Input

The input file contains two integers n and M, $1 \leqslant n \leqslant 1\,000$, $1 \leqslant M \leqslant 10^{18}$.

Output

Output one integer — the number of losing positions modulo $10^9 + 7$.

Examples

meganim.in	meganim.out
3 4	19

Problem G. Paratroopers

Input file:	`paratroopers.in`
Output file:	`paratroopers.out`
Time limit:	3 seconds
Memory limit:	256 megabytes

Andrew has sent n paratroopers on a super-secret mission to the Grand Canyon. Each paratrooper has landed somewhere in the canyon, but it's hard to tell where exactly.

For simplicity and formality, we will assume that the Grand Canyon is a line segment of unit length. i-th paratrooper has landed in point a_i, $0 < a_i < 1$. We don't know a_i, all we know is that they are all different.

Before the mission can commence, Andrew must find out where exactly his paratroopers are — that is, he needs to know a_i. What he can do is to call any paratrooper via mobile phone, and this paratrooper will tell Andrew what does he see to the left (either the end of the canyon or another paratrooper, in which case Andrew will know which paratrooper is that) and how far away that object is, and what does he see to the right and how far away that object is. Left is the direction towards point 0 of the canyon, and right is the direction towards point 1 of the canyon. Andrew can then call another paratrooper, and so on until he can determine all paratroopers' locations.

For example, suppose there are three paratroopers, and they have landed in points 0.2, 0.7, 0.5 respectively. First, Andrew calls the second paratrooper, and he reports that he sees the third paratrooper at distance 0.2 to his left, and the wall of the canyon at distance 0.3 to his right. He can then deduce that the location of the second paratrooper is 0.7 (0.3 to the left of the right wall which is 1), and the location of the third paratrooper is 0.5 (0.2 to the left of 0.7). Then, Andrew calls the first paratrooper, which reports that he sees the third paratrooper at distance 0.3 to his right, and the wall of the canyon at distance 0.2 to his left. Now Andrew can determine the locations of all three paratroopers (in fact, he even has double evidence for the location of the third paratrooper).

Andrew wants to minimize the number of phone calls being made, as the mission is super-secret. However, he hasn't planned his actions in advance, and has already made some calls, so all he wants is to minimize the number of remaining calls.

How many calls will he need to make in the worst case if he follows the best possible strategy?

Input

The first line of the input file contains the number of paratroopers n ($1 \leqslant n \leqslant 100\,000$) and the number of phone calls already made k ($1 \leqslant k \leqslant 50\,000$). The next k lines contain paratroopers' replies to previous phone calls. Each of those lines contain 5 numbers a, id_l, d_l, id_r, d_r. a ($1 \leqslant a \leqslant n$) is the number of the paratrooper that was called, id_l ($0 \leqslant id_l \leqslant n$) is the number of the paratrooper to its left (or 0 if there's a wall to its left), d_l ($0 < d_l < 1$) is the floating-point distance to the left object, id_r ($0 \leqslant id_r \leqslant n$) and d_r ($0 < d_r < 1$) describe the right side in the same manner. The distances have at most 8 digits after the decimal point.

It is guaranteed that there exists at least one possible valid position of all paratroopers that will result in such replies to phone calls. Andrew never calls one paratrooper more than once.

Output

In the only line of the output file, write how many more calls will Andrew need to make in the worst case to determine the locations of all paratroopers.

Examples

paratroopers.in	paratroopers.out
3 1 2 3 0.2 0 0.3	1

Problem H. Random Substring

Input file:	`random-substring.in`
Output file:	`random-substring.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Given an alphabet with n letters, and two lengths $l_1 \leqslant l_2$, what is the probability that a random string with l_1 letters is a substring of a random string with l_2 letters?

In a random string, each letter is chosen uniformly and independently from all letters of the alphabet.

Your answer will be accepted when it's within 10^{-9} of the correct answer.

Input

The input file contains three integers n, l_1, l_2 ($2 \leqslant n \leqslant 100$, $1 \leqslant l_1 \leqslant l_2 \leqslant 100$).

Output

Output one floating-point number — the sought probability. Your answer will be accepted when it's within 10^{-9} of the correct answer.

Examples

random-substring.in	random-substring.out
`3 2 3`	`0.20987654320987653`

Problem I. Wrapping Threads Around

Input file:	`wrap-around.in`
Output file:	`wrap-around.out`
Time limit:	3 seconds
Memory limit:	256 megabytes

Andrew has many threads — long cotton threads. Each thread's length is an integer amount of meters, and each meter is colored with some color. For example, a 3-meter thread might be colored like this: 1 meter red, then 1 meter green, then 1 meter blue.

Moreover, all Andrew's threads are colored similarly: we can number all colors from 1 to k in such a way that 1 meter of color $i < k$ is always followed by 1 meter of color $i + 1$, and 1 meter of color k is always followed by 1 meter of color 1. For example, suppose $k = 3$, color red is number 1, color green is number 2, and color blue is number 3. Then, for example, a thread might be colored like this: 1 meter green, then 1 meter blue, then 1 meter red, then 1 meter green, then 1 meter blue. We can denote that thread as GBRGB (using first letters of each color's name). The following threads are also possible: RGB, RGBRGBRG, GB, R, BR. However, the following threads are impossible: RR, BGR, RGBGB.

Andrew wants to assemble one long thread from his threads. He can tie them together at ends, and he can also cut them in the middle (only in integer points — that is, between different colors; after a thread has been cut into several parts, he can do whatever he wants with each part — use to assemble the long thread, or just throw away). He wants the long thread to also follow the same color pattern.

For example, suppose he has threads RGBRGBR, R, RG and RGB. He can cut thread RGBRGBR in two: RGBRGB and R, and then assemble the following long thread: RGB+RGBRGB+RG=RGBRGBRGBRG, while throwing two R threads away.

Andrew wants to avoid cutting existing threads too much. More specifically, he is willing to cut each thread he has into at most three (3) parts. What is the longest thread he can then assemble?

Input

The first line of the input file contains two integers: n ($1 \leqslant n \leqslant 50\,000$), denoting the number of Andrew's threads, and k ($3 \leqslant k \leqslant 50\,000$), denoting the number of different colors.

The next n lines contain two integers each, describing the threads: c ($1 \leqslant c \leqslant k$), denoting the color of the first meter of a thread, and l ($1 \leqslant l \leqslant 1\,000\,000\,000$), denoting the length of the thread.

Output

On the first line of the output file, describe the longest thread Andrew can assemble: print the color of its first meter and its length.

On the next n lines, describe how you cut existing threads. i-th line should describe the result of cutting i-th thread from the input file in the following format: first integer m ($1 \leqslant m \leqslant 3$) is the number of parts the thread is being cut into, then m integers follow, describing the lengths of the parts in the order they appear along the thread.

On the last line, describe how you assemble the long thread. First integer m is the number of parts the thread is assembled from, then m integers follow, describing the lengths of the parts in the order they appear along the thread.

In case there are several possible optimal solutions, output any.

Examples

wrap-around.in	wrap-around.out

3 4
2 2
3 3
3 2
	3 6
	1 2
	1 3
	2 1 1
	3 3 2 1

Note

Here's what happens in the example case. There are 4 colors, let's name them A, B, C and D. We have the following threads initially: BC, CDA, CD. We cut CD into C and D, and then assemble CDA+BC+D (throwing C away).