Codeforces Beta Round #35 (Div. 2)

A. Shell Game

time limit per test: 2 seconds
memory limit per test: 64 megabytes
input: input.txt
output: output.txt

Today the «Z» city residents enjoy a shell game competition. The residents are gathered on the main square to watch the breath-taking performance. The performer puts 3 non-transparent cups upside down in a row. Then he openly puts a small ball under one of the cups and starts to shuffle the cups around very quickly so that on the whole he makes exactly 3 shuffles. After that the spectators have exactly one attempt to guess in which cup they think the ball is and if the answer is correct they get a prize. Maybe you can try to find the ball too?

Input
The first input line contains an integer from 1 to 3 — index of the cup which covers the ball before the shuffles. The following three lines describe the shuffles. Each description of a shuffle contains two distinct integers from 1 to 3 — indexes of the cups which the performer shuffled this time. The cups are numbered from left to right and are renumbered after each shuffle from left to right again. In other words, the cup on the left always has index 1, the one in the middle — index 2 and the one on the right — index 3.

Output
In the first line output an integer from 1 to 3 — index of the cup which will have the ball after all the shuffles.

Examples

input
1 1 2 2 1 2 1
output
2

input
1 2 1 3 1 1 3
output
2

B. Warehouse

time limit per test: 2 seconds
memory limit per test: 64 megabytes
input: input.txt
output: output.txt

Once upon a time, when the world was more beautiful, the sun shone brighter, the grass was greener and the sausages tasted better Arlandia was the most powerful country. And its capital was the place where our hero DravDe worked. He couldn't program or make up problems (in fact, few people saw a computer those days) but he was nevertheless happy. He worked in a warehouse where a magical but non-alcoholic drink Ogudar-Olok was kept. We won't describe his work in detail and take a better look at a simplified version of the warehouse.

The warehouse has one set of shelving. It has n shelves, each of which is divided into m sections. The shelves are numbered from top to bottom starting from 1 and the sections of each shelf are numbered from left to right also starting from 1. Each section can contain exactly one box of the drink, and try as he might, DravDe can never put a box in a section that already has one. In the course of his work DravDe frequently notices that he has to put a box in a filled section. In that case his solution is simple. DravDe ignores that section and looks at the next one to the right. If it is empty, he puts the box there. Otherwise he keeps looking for the first empty section to the right. If no empty section is found by the end of the shelf, he looks at the shelf which is under it, then the next one, etc. Also each time he looks at a new shelf he starts from the shelf's beginning. If DravDe still can't find an empty section for the box, he immediately drinks it all up and throws the empty bottles away not to be caught.

After one great party with a lot of Ogudar-Olok drunk DravDe asked you to help him. Unlike him, you can program and therefore modeling the process of counting the boxes in the warehouse will be easy work for you.

The process of counting contains two types of query messages:

- «**+1 x y id**» (where x, y are integers, $1 \le x \le n$, $1 \le y \le m$, and id is a string of lower case Latin letters — from 1 to 10 characters long). That query means that the warehouse got a box identified as id, which should be put in the section y on the shelf x. If the section is full, use the rules described above. It is guaranteed that every moment of the process the identifiers of all the boxes in the warehouse are different. You don't have to answer this query.
- «**-1 id**» (where id is a string of lower case Latin letters — from 1 to 10 characters long). That query means that a box identified as id is removed from the warehouse. You have to answer this query (see output format).

Input

The first input line contains integers n, m and k ($1 \le n, m \le 30$, $1 \le k \le 2000$) — the height, the width of shelving and the amount of the operations in the warehouse that you need to analyze. In the following k lines the queries are given in the order of appearance in the format described above.

Output

For each query of the «**-1 id**» type output two numbers in a separate line — index of the shelf and index of the section where the box with this identifier lay. If there was no such box in the warehouse when the query was made, output «**-1 -1**» without quotes.

Examples

input
2 2 9 +1 1 1 cola +1 1 1 fanta +1 1 1 sevenup +1 1 1 whitekey -1 cola -1 fanta -1 sevenup -1 whitekey -1 cola
output
1 1 1 2 2 1 2 2 -1 -1

input
2 2 8 +1 1 1 cola -1 cola +1 1 1 fanta -1 fanta +1 1 1 sevenup -1 sevenup +1 1 1 whitekey -1 whitekey
output

```
1 1
1 1
1 1
1 1
```

C. Fire Again

time limit per test: 2 seconds
memory limit per test: 64 megabytes
input: input.txt
output: output.txt

After a terrifying forest fire in Berland a forest rebirth program was carried out. Due to it N rows with M trees each were planted and the rows were so neat that one could map it on a system of coordinates so that the j-th tree in the i-th row would have the coordinates of (i, j). However a terrible thing happened and the young forest caught fire. Now we must find the coordinates of the tree that will catch fire last to plan evacuation.

The burning began in K points simultaneously, which means that initially K trees started to burn. Every minute the fire gets from the burning trees to the ones that aren't burning and that the distance from them to the nearest burning tree equals to 1.

Find the tree that will be the last to start burning. If there are several such trees, output any.

Input

The first input line contains two integers N, M ($1 \le N, M \le 2000$) — the size of the forest. The trees were planted in all points of the (x, y) ($1 \le x \le N, 1 \le y \le M$) type, x and y are integers.

The second line contains an integer K ($1 \le K \le 10$) — amount of trees, burning in the beginning.

The third line contains K pairs of integers: $x_1, y_1, x_2, y_2, \ldots, x_k, y_k$ ($1 \le x_i \le N, 1 \le y_i \le M$) — coordinates of the points from which the fire started. It is guaranteed that no two points coincide.

Output

Output a line with two space-separated integers x and y — coordinates of the tree that will be the last one to start burning. If there are several such trees, output any.

Examples

input
3 3 1 2 2
output
1 1

input
3 3 1 1 1
output
3 3

input
3 3 2 1 1 3 3
output
2 2

D. Animals

time limit per test: 2 seconds
memory limit per test: 64 megabytes
input: input.txt
output: output.txt

Once upon a time DravDe, an outstanding person famous for his professional achievements (as you must remember, he works in a warehouse storing Ogudar-Olok, a magical but non-alcoholic drink) came home after a hard day. That day he had to drink 9875 boxes of the drink and, having come home, he went to bed at once.

DravDe dreamt about managing a successful farm. He dreamt that every day one animal came to him and asked him to let it settle there. However, DravDe, being unimaginably kind, could send the animal away and it went, rejected. There were exactly n days in DravDe's dream and the animal that came on the i-th day, ate exactly c_i tons of food daily starting from day i. But if one day the animal could not get the food it needed, it got really sad. At the very beginning of the dream there were exactly X tons of food on the farm.

DravDe woke up terrified...

When he retold the dream to you, he couldn't remember how many animals were on the farm by the end of the n-th day any more, but he did remember that nobody got sad (as it was a happy farm) and that there was the maximum possible amount of the animals. That's the number he wants you to find out.

It should be noticed that the animals arrived in the morning and DravDe only started to feed them in the afternoon, so that if an animal willing to join them is rejected, it can't eat any farm food. But if the animal does join the farm, it eats daily from that day to the n-th.

Input

The first input line contains integers n and X ($1 \le n \le 100, 1 \le X \le 10^4$) — amount of days in DravDe's dream and the total amount of food (in tons) that was there initially. The second line contains integers c_i ($1 \le c_i \le 300$). Numbers in the second line are divided by a space.

Output

Output the only number — the maximum possible amount of animals on the farm by the end of the n-th day given that the food was enough for everybody.

Examples

input
3 4 1 1 1
output
2

input
3 6 1 1 1
output
3

Note

Note to the first example: DravDe leaves the second and the third animal on the farm. The second animal will eat one ton of food on the second day and one ton on the third day. The third animal will eat one ton of food on the third day.

E. Parade

time limit per test: 4 seconds
memory limit per test: 64 megabytes
input: input.txt
output: output.txt

No Great Victory anniversary in Berland has ever passed without the war parade. This year is not an exception. That's why the preparations are on in full strength. Tanks are building a line, artillery mounts are ready to fire, soldiers are marching on the main square... And the air forces general Mr. Generalov is in trouble again. This year a lot of sky-scrapers have been built which makes it difficult for the airplanes to fly above the city. It was decided that the planes should fly strictly from south to north. Moreover, there must be no sky scraper on a plane's route, otherwise the anniversary will become a tragedy. The Ministry of Building gave the data on n sky scrapers (the rest of the buildings are rather small and will not be a problem to the planes). When looking at the city from south to north as a geometrical plane, the i-th building is a rectangle of height h_i. Its westernmost point has the x-coordinate of l_i and the easternmost — of r_i. The terrain of the area is plain so that all the buildings stand on one level. Your task as the Ministry of Defence's head programmer is to find an *enveloping* polyline using the data on the sky-scrapers. The polyline's properties are as follows:

- If you look at the city from south to north as a plane, then any part of any building will be inside or on the boarder of the area that the polyline encloses together with the land surface.
- The polyline starts and ends on the land level, i.e. at the height equal to 0.
- The segments of the polyline are parallel to the coordinate axes, i.e. they can only be vertical or horizontal.
- The polyline's vertices should have integer coordinates.
- If you look at the city from south to north the polyline (together with the land surface) must enclose the minimum possible area.
- The polyline must have the smallest length among all the polylines, enclosing the minimum possible area with the land.
- The consecutive segments of the polyline must be perpendicular.

Picture to the second sample test (the enveloping polyline is marked on the right).

Input

The first input line contains integer n ($1 \le n \le 100000$). Then follow n lines, each containing three integers h_i, l_i, r_i ($1 \le h_i \le 10^9$, $-10^9 \le l_i < r_i \le 10^9$).

Output

In the first line output integer m — amount of vertices of the enveloping polyline. The next m lines should contain 2 integers each — the position and the height of the polyline's vertex. Output the coordinates of each vertex in the order of traversing the polyline from west to east. Remember that the first and the last vertices of the polyline should have the height of 0.

Examples

input
2 3 0 2 4 1 3
output
6 0 0 0 3 1 3 1 4 3 4 3 0

input
5 3 -3 0 2 -1 1 4 2 4 2 3 7 3 6 8
output
14 -3 0 -3 3 0 3 0 2 1 2 1 0 2 0 2 4 4 4 4 2 6 2 6 3

```
8 3
8 0
```

Codeforces (c) Copyright 2010-2016 Mike Mirzayanov
The only programming contests Web 2.0 platform