Mega Inversions [Kattis - megainversions]

The n^2 upper bound for any sorting algorithm is easy to obtain: just take two elements that are misplaced with respect to each other and swap them. Conrad conceived an algorithm that proceeds by taking not two, but *three* misplaced elements. That is, take three elements $a_i > a_j > a_k$ with $i < j < k$ and place them in order a_k, a_j, a_i. Now if for the original algorithm the steps are bounded by the maximum number of inversions $\frac{n(n-1)}{2}$, Conrad is at his wits' end as to the upper bound for such triples in a given sequence. He asks you to write a program that counts the number of such triples.

Input

The first line of the input is the length of the sequence, $1 \leq n \leq 10^5$. The next line contains the integer sequence $a_1, a_2, \ldots, a_n$. You can assume that all $a_i \in [1, n]$.

Output

Output the number of inverted triples.

Sample 1

Input	copy	Output	copy
3 1 2 3		0	

Sample 2

Input	copy	Output	copy
4 3 3 2 1		2	

Overlapping Rectangles 计蒜客 - A1282

There are n rectangles on the plane. The problem is to find the area of the union of these rectangles. Note that these rectangles might overlap with each other, and the overlapped areas of these rectangles shall not be counted more than once. For example, given a rectangle A with the bottom left corner located at $(0, 0)$ and the top right corner at $(2, 2)$, and the other rectangle B with the bottom left corner located at $(1, 1)$ and the top right corner at $(3, 3)$, it follows that the area of the union of A and B should be 7, instead of 8.

Although the problem looks simple at the first glance, it might take a while to figure out how to do it correctly. Note that the shape of the union can be very complicated, and the intersected areas can be overlapped by more than two rectangles.

Note:

(1) The coordinates of these rectangles are given in integers. So you do not have to worry about the floating point round-off errors. However, these integers can be as large as $1,000,000$.

(2) To make the problem easier, you do not have to worry about the sum of the areas exceeding the long integer precision. That is, you can assume that the total area does not result in integer overflow.

Input Format

Several sets of rectangles configurations. The inputs are a list of integers. Within each set, the first integer (in a single line) represents the number of rectangles, n, which can be as large as 1000. After n, there will be n lines representing the n rectangles; each line contains four integers $< a, b, c, d >$, which means that the bottom left corner of the rectangle is located at (a, b), and the top right corner of the rectangle is located at (c, d). Note that integers a, b, c, d can be as large as $1,000,000$.

These configurations of rectangles occur repetitively in the input as the pattern described above. An integer $n = 0$ (zero) signifies the end of input.

Output Format

For each set of the rectangles configurations appeared in the input, calculate the total area of the union of the rectangles. Again, these rectangles might overlap each other, and the intersecting areas of these rectangles can only be counted once. Output a single star '*' to signify the end of outputs.

Sample 1

Input	copy	Output	copy
2 0 0 2 2 1 1 3 3 3 0 0 1 1 2 2 3 3 4 4 5 5 0		7 3 *	

Supercomputer Kattis - supercomputer

Jóhann, Marteinn and Símon have decided to make
the next generation of supercomputers! They know that it probably won't be long before Quantum computers take over, but since they don't know anything about Quantum mechanics, they want to rush these new supercomputers out into the market, make their money, and hopefully retire with their wealth.

Since they're trying to sell these things, they decide they need some cool features to promote the computers. Marteinn suggests painting flames on the back of the computers to make it look like they're computing faster. Jóhann agrees, but suggests also adding a second keyboard so people can type faster, just like in that TV show: NCIS. Símon also agrees, but he thinks there's something missing. What are they forgetting? Ah, of course, faster memory!

They decide to add an N-bit memory that supports the following two operations:

- given an integer k, flip the k:th bit of the memory (changing a 0 to a 1, and a 1 to a 0), and
- given integers l and r, count how many bits between the l:th bit and the r:th bit are set to 1.

After announcing their new supercomputer with these three awesome features, they immediately received a large amount of orders. Of course everyone wants a supercomputer with flames painted on the back! But now it's time to actually implement these features. While Jóhann, Marteinn and Símon are busy painting the computers and supplying more keyboards, they've hired you to implement their memory.

Input

The input consists of:

- one line containing two integers N ($1 \leq N \leq 10^6$), the number of bits in the

memory, and K ($1 \leq K \leq 10^5$), the number of queries;

- K lines each of the form:
  - `F` i ($1 \leq i \leq N$) representing a query to flip the i:th bit in memory;
  - `C` l r ($1 \leq l \leq r \leq N$) representing a query to count the number of 1 bits in the range from l to r, inclusive.

All N bits in the memory are initially set to 0.

Output

For each query of the second type, output a single line containing the number of bits set to 1 in the given range.

Sample 1

Input	copy	Output	copy
6 7 F 3 C 2 5 F 3 F 4 F 5 C 2 5 C 1 4		1 2 1	

Find my Family

Kattis - findmyfamily

You are looking for a particular family photo with you and your favorite relatives Alice and Bob. Each family photo contains a line-up of n people. On the photo you're looking for, you remember that Alice, who is taller than you, was somewhere on your left from the perspective of the photographer. Also, Bob who is taller than both you and Alice, was standing somewhere on your right.

Since you have a large number of family photos, you want to use your computer to assist in finding the photo. Many of the photos are quite blurry, so facial recognition has proven ineffective. Luckily, the Batch Apex Photo Classifier, which detects each person in a photo and outputs the sequence of their (distinct) heights in pixels, has produced excellent results. Given this sequence of heights for k photos, determine which of these photos could potentially be the photo you're looking for.

Input

- The first line contains $1 \leq k \leq 1\,000$, the number of photos you have to process.
- Then follow two lines for each photo.
    - The first line contains a single integer $3 \leq n \leq 3 \cdot 10^5$, the number of people on this photo.
    - The second line contains n distinct integers $1 \leq h_1, \ldots, h_n \leq 10^9$, the heights of the people in the photo, from left to right.

It is guaranteed that the total number of people in all photos is at most $3 \cdot 10^5$.

Output

- On the first line, output the number of photos k that need further investigation.
- Then print k lines each containing a single integer $1 \leq a_i \leq n$, the sorted indices

of the photos you need to look at.

Sample 1

Input	copy	Output	copy
1 3 2 1 3		1 1	

Sample 2

Input	copy	Output	copy
4 4 140 157 160 193 5 15 24 38 9 30 6 36 12 24 29 23 15 6 170 230 320 180 250 210		2 2 4	

Antimatter Rain

Kattis - antimatterrain

You've heard of acid rain but have you heard of antimatter rain? Antimatter rain is so potent that when it comes into contact with another object, it immediately disintegrates both itself and the object. Kayla's job as a SpaceFleet Researcher is gathering weather data on exotic planets. This time, their assignment is to monitor the antimatter rainfall.

Sensors are set up in the planet's atmosphere and are about to be rained on with antimatter rain. Oh no! Kayla monitors a single 2D section. Each sensor is either a single horizontal strip or a single point. When one or more antimatter droplet fall on a single sensor, all of those droplets and the sensor disintegrate simultaneously. That is, they disappear. All other droplets will drop past where the sensor used to be.

Kayla sees all the antimatter rain drops the moment before they all start to fall. All droplets fall at exactly the same rate.

For each droplet, Kayla wants to know if and where it will disintegrate. Help them out with this demanding task!

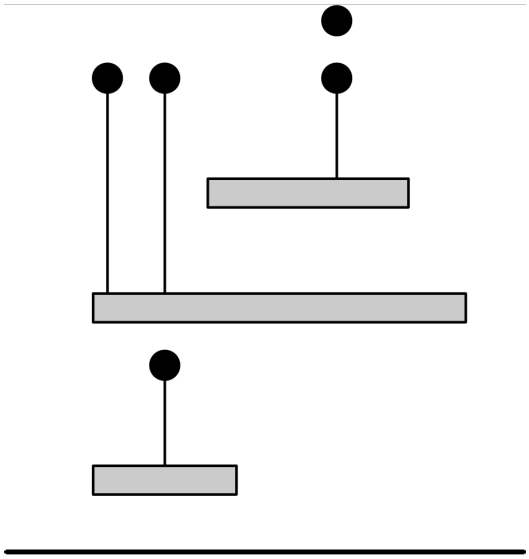

Illustration of the first sample. The vertical lines connect the drops to the sensor they hit. The drop with no associated vertical line will not hit any sensor.

Input

The first line of input contains two integers D ($1 \leq D \leq 100\,000$), which is the number of antimatter droplets, and S ($1 \leq S \leq 100\,000$), which is the number of sensors.

The next D lines describe the droplets, in order. Each of these lines contains two integers x ($1 \leq x \leq 10^9$), which is the x-coordinate of the droplet and y ($1 \leq y \leq 10^9$), which is the y-coordinate of the droplet.

The next S lines describe the sensors. Each line contains three integers x_1, x_2 ($1 \leq x_1 \leq x_2 \leq 10^9$), which is the leftmost and the rightmost x-coordinate of the sensor, and y ($1 \leq y \leq 10^9$), which is the y-coordinate of the sensor.

It is guaranteed that no two drops will start in the same location, no drop will start on any sensor, and no two sensors touch (not even at a single point).

Output

For each droplet, in order, display a single number indicating the y-coordinate that it will disintegrate. If the droplet does not disintegrate, display 0 instead. These values should appear on separate lines.

Sample 1

Input	Output
5 3 1 8 2 3 2 8 5 8 5 9 3 6 6 1 7 4 1 3 1	4 1 4 6 0

Sample 2

Input	copy	Output	copy
6 3 1 2 4 8 5 10 6 10 7 10 8 10 1 1 1 3 4 3 5 7 9		1 3 9 9 9 0	

Toll [Kattis - toll](#)

A trucking company wants to optimize its internal processes—which mainly means saving money. The company serves a region where a toll must be paid for every single street. Each street connects two places (cities, villages etc.) directly. The company serves a set of orders; each order tells them to carry goods from one place to another. When serving an order, the company wants to pay the minimum overall toll. As the region's street network can be modeled by a graph where each edge has a specific cost (the toll for the respective street), the company actually wants to know (the cost of) the cheapest path between two nodes in this graph.

However, the region's street network graph has an interesting property: it is directed (i.e. all streets are oneway), and there can only be an edge from a to b if $\lfloor b/K \rfloor = 1 + \lfloor a/K \rfloor$ (for some constant K).

Write a program that for each of a given list of orders outputs the minimum toll the company has to pay to serve the respective order.

Input

The first line contains four integers: K (with the meaning described above), N (the number of places), M (the number of streets), and O (the number of orders).

Each of the next M lines contains three integers a, b, t ($0 \leq a, b < N$). This means that there is a (oneway) street from a to b with toll t. You are guaranteed that $\lfloor b/K \rfloor = 1 + \lfloor a/K \rfloor$ is satisfied, and that no two locations are connected by more than one street.

Finally O lines follow, each containing two integers a, b: this means that there is an order to carry goods from place a to place b.

We always have $1 \leq N \leq 50\,000$, $1 \leq O \leq 10\,000$ and $1 \leq K \leq 5$. Moreover, we have $0 \leq a < b < N$ for all orders a, b and $1 \leq t \leq 10\,000$ for all tolls t.

Output

Your output should consist of O lines, each with one integer. The i-th line should contain the toll on a cheapest path between the two places in order i. If no such path exists, output -1 in this line.

Sample 1

Input	copy	Output	copy
5 14 5 5 0 5 9 5 12 10 0 7 7 7 12 8 4 7 10 0 12 0 5 0 7 7 12 0 13		15 9 7 8 -1	