Problem A. Oceanic Battle

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2.5 seconds
Memory limit:	256 megabytes

The playfield of the game "Oceanic Battle" is grid $M \times N$ cells. We will consider first stage — the ships placement. In this game ship is represented by the rectangles built from the grid cells. Value of the ship is number of cells in the rectangle. Ships cannot intersect or touch, i.e. no two ships have common point.

Given partial ships arrangement (several ships placed on the playfield following game rules), find out maximal possible value of single ship in the final arrangement, if it is allowed to add any number of extra ships (while it is possible to put them without breaking game rules).

Input

First line of the input contains two integers M and N — dimensions of the playfield ($1 \le M, N \le 5000$). Next line contains one integer K ($0 \le K \le 10^4$) — number of ships already placed. Each of the next K lines contains four integers X_1, Y_1, X_2, Y_2 — coordinates of opposite vertices of the rectangle, representing some ship ($0 \le X_1, X_2 \le M$, $0 \le Y_1, Y_2 \le N$, $X_1 \ne X_2$, $Y_1 \ne Y_2$). You may assume that the ship arrangement in the input is correct, i.e. no two ships have common point.

Output

Print one integer — maximal possible value of ship in the arrangement which includes all the given ships at the same places.

Examples

standard input	standard output

4 4
1
0 0 1 2
	8

7 3
2
1 3 2 0
6 0 7 1
	6

Problem B. K-th Maximum

Input file:	`kthmax.in`
Output file:	`kthmax.out`
Time limit:	1 second
Memory limit:	256 megabytes

Implement a data structure that allows to add and delete elements, and also answers the k-th maximum element queries.

Input

The first line of the input contains a single integer n ($1 \le n \le 100\,000$) — the number of queries to perform. Then follow n lines containing the queries. Every query is given by two numbers c_i and k_i ($|k_i| \le 10^9$) — the type and the value of the corresponding query. The possible types are:

- $+1$ (or just 1): add an element with the key equal to k_i.
- 0: find and print k_i-th maximum.
- -1: delete an element with the key equal to k_i.

It guaranteed, that during the workflow the structure will never be asked to store two elements with equal keys or to delete an element that is not present in the structure. It's guaranteed that the answer for every query of k_i-th maximum exists.

Output

For every query of the 0 type print the line containing a single integer — k_i-th maximum.

Examples

kthmax.in	kthmax.out
11 +1 5 1 3 +1 7 0 1 0 2 0 3 -1 5 +1 10 0 1 0 2 0 3	7 5 3 10 7 3

Problem C. Swapper

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

Your goal in this problem is to implement the data structure called swapper. It should be able to perform the following operations:

- Take some segment of even length from x to y and swap elements x and $x+1$, $x+2$ and $x+3$, and so on.
- Count the sum of elements on some segment from a to b.

Input

The input contains of one or more test data.

First line of each test starts with two integers n and m ($1 \le n, m \le 100\,000$) — the length of the sequence and the number of queries. The second line contains n integers not exceeding 10^6 by their absolute value. Then follow m lines describing the queries: 1 x_i y_i — query of the first type; 2 a_i b_i — query of the second type. The sum of n in one input doesn't exceed $200\,000$. Also, the sum of m in one input doesn't exceed $200\,000$. Input ends with line containing two zeroes. This line shouldn't be processed as a test. It's guaranteed that $x_i < y_i$ and $a_i \le b_i$.

Output

For each test print the answer for each query. Follow the format of the sample. Separate two tests with an empty line.

Examples

swapper.in	swapper.out

5 5
1 2 3 4 5
1 2 5
2 2 4
1 1 4
2 1 3
2 4 4
0 0
	Swapper 1:
	10
	9
	2

Problem D. Range Minimum Query

Input file:	`rmq.in`
Output file:	`rmq.out`
Time limit:	1 second
Memory limit:	256 mebibytes

Company *Giggle* plans to open a new office in Sudislavl and you are invited for an interview. You are asked to solve the following problem.

Implement the data structure that holds an array of integers. Initially, the array is empty. You have to process queries of two types:

- "`? i j`" — return the minimum element between i and j inclusive.
- "`+ i x`" — add integer x right after the i-th element of array. If $i = 0$, then the new element is added at the beginning.

Of course, this structure should work pretty fast.

Input

First line of the input contains a single integer n — the number of operations to process ($1 \le n \le 200\,000$). Then follow n lines with queries descriptions. All operations are guaranteed to be valid. All integers of array do not exceed 10^9 by their absolute value.

Output

For each operation, print the result in a separate line.

Examples

rmq.in	rmq.out
8 + 0 5 + 1 3 + 1 4 ? 1 2 + 0 2 ? 2 4 + 4 1 ? 3 5	4 3 1

Problem E. Permutations Strike Back

Input file:	`permutation2.in`
Output file:	`permutation2.out`
Time limit:	3 seconds
Memory limit:	256 megabytes

Vasya has written n integers from 1 to n on the blackboard. Sometimes he changes the integers written on some positions. You have to write the program that will support this change operation and will also be able to answer the query: how many integer on position from x to y are in range from k to l.

Input

The first line of the input contains two integers n and m ($1 \le n, m \le 100\,000$) — the number of integers and the total number of queries Vasya wants you to process.

The second line contains n integers — the initial sequence, written by Vasya on the blackboard. Them follow m lines describing the queries. Each change query starts with the word `SET` and has form `SET a b` ($1 \le a \le n$, $1 \le b \le n$), meaning that Vasya changes the integer at position a to integer b. Each count query starts with the word `GET` and has form `GET x y k l` ($1 \le x \le y \le n$, $1 \le k \le l \le n$)

Output

Print the answers to count queries in order they appear in the input.

Examples

permutation2.in	permutation2.out
4 4 1 2 3 4 GET 1 2 2 3 GET 1 3 1 3 SET 1 4 GET 1 3 1 3	1 3 2

Problem F. Amber Ball

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

Amber ball falls down on the ground facing some obstacles. Introduce standard Cartesian coordinates such that the ground is an Ox axis and the gravity forces the ball to go in negative direction of Oy axis. Obstacles are segments and the ball is a point. If the ball faces an obstacle (even its highest point) it rolls down to the lowest point of an obstacle and then continues to fall down vertically (in other words, the horizontal component of its movement disappears immediately). There are no vertical and no horizontal obstacles, they all are located strictly above the ground and no two obstacles have a common point. Thus, the ball will eventually reach the ground at some point.

The ball is being thrown multiple times from different starting positions. For each starting position you have to determine the x coordinates of the point where the ball will touch the ground.

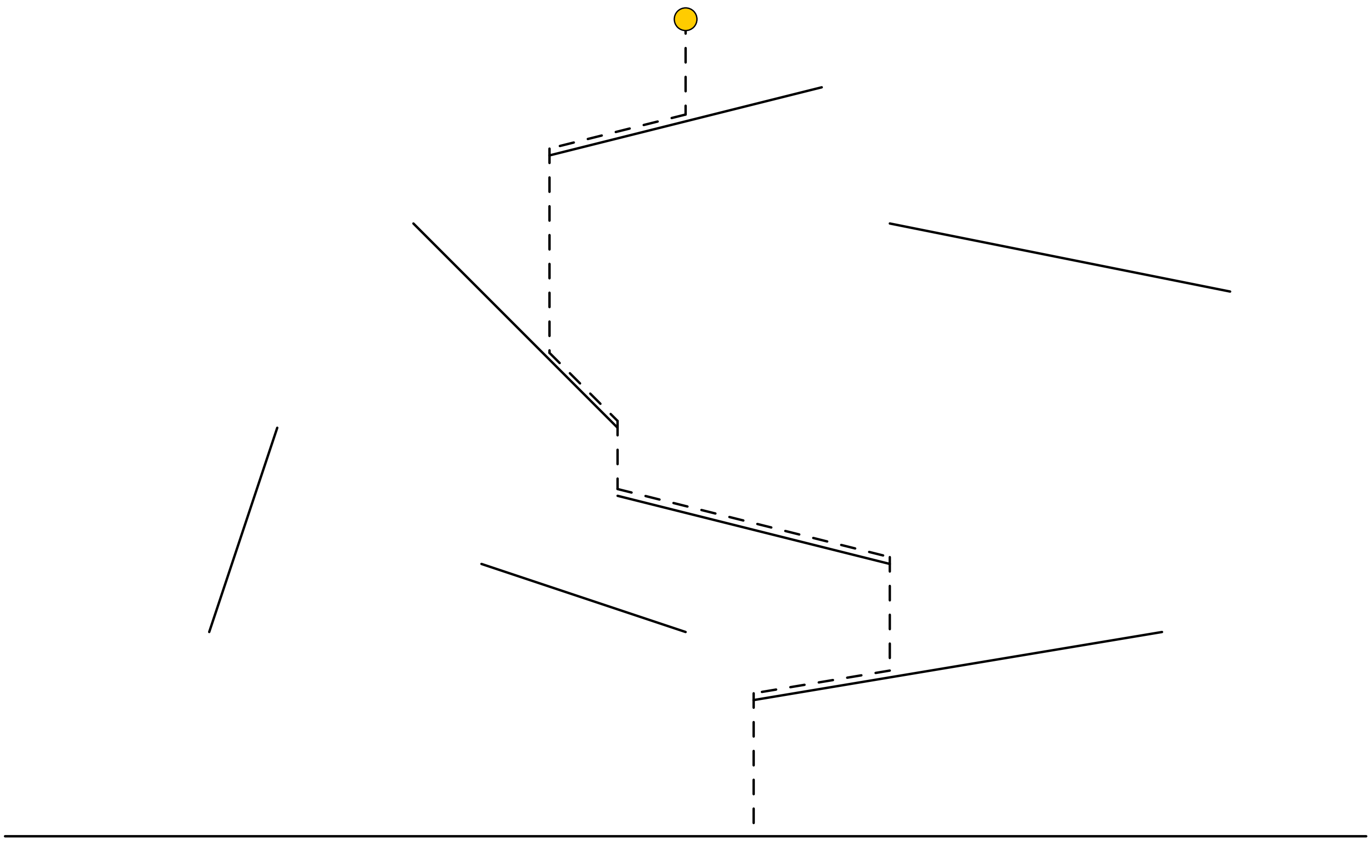

Figure 1: *
Third sample.

Input

The first line of the input contains a single integer n ($0 \leq N \leq 3 \cdot 10^5$) — the number of obstacles.

Each of the next n lines contains the description of one obstacle — four integers x_1, y_1, x_2, y_2 ($x_1 < x_2$, $y_1 \neq y_2$, $y_1, y_2 > 0$): (x_1, y_1) — coordinates of the left end, (x_2, y_2) — coordinates of the right end.

Next line contains the only integer m ($1 \leq m \leq 3 \cdot 10^5$) — the number of experiments.

Then follow m lines, each of them contains a single integer — x-coordinate of the starting position. You may assume that the y coordinate of the starting position is bigger than y-coordinate of any point of any obstacle.

All coordinates in the input are integers not exceeding 10^6 by their absolute value. It's guaranteed that there are no vertical or horizontal obstacles and no two obstacles share a common point. The length of each obstacle is positive.

Output

For each starting position print one integer — the x-coordinate of the point where the ball will touch the ground.

Examples

a.in	a.out

2
0 7 1 3
3 3 4 7
2
2
4
	2
	3

2
-3 5 1 3
-1 1 1 2
3
-3
-4
1
	-1
	-4
	-1

7
-2 10 2 11
-4 9 -1 6
3 9 8 8
-7 3 -6 6
-1 5 3 4
-3 4 0 3
1 2 7 3
1
0
	1

Note

The picture above illustrates the third sample.

Problem G. Database

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	256 megabytes

Head teacher of the Byteland school issued an order to control students' academic performance: each student is required to provide information about his grades at the end of each week. However, it is required to provide only the average grade, which is believed to completely describe the student's performance during the week.

Parents like this innovation, because now they can monitor the academic performance of their children and compare it to the performance of other students. Every Saturday students submit their average grades to the school database where they are stored. After that, parents perform m queries.

Let u be the maximum grade stored in the database at the current time. Let us denote as $cnt(x)$ the number of grades stored in the database, that are greater than or equal to x (there can be equal grades, each one is counted). Database supports four types of queries:

1. Replace all numbers stored in the database with the sequence $(cnt(1), cnt(2), \ldots, cnt(u))$.
2. Add integer x to the database.
3. Remove from the database a single occurrence of integer x (if there is at least one).
4. Count the number of elements equal to x.

Parents begin to query information they need and update the data only after all n students send their grades to the database.

Unfortunately, the school license for this database has just expired, so now you have to perform all the operations manually.

Input

The first line of the input contains two integers n and m ($1 \le n, m \le 200\,000$) — the number of students and the number of queries to perform, respectively.

The second line contains n integers g_i ($1 \le g_i \le 200\,000$) — the average grades of the students.

The following m lines describe the queries to the database in order they should be processed. Each description starts with one of the characters '`t`', '`a`', '`r`' or '`c`', meaning the query is of the first, the second, the third, or the fourth type, respectively. If the query is of the second, the third or the fourth type, the character is followed by an integer x_i ($1 \le x_i \le 200\,000$) — query parameter.

Output

First print the answers to all queries of the fourth type in order they appear in the input file. Then print in non-decreasing order all integers stored in the database after all queries are performed.

It's guaranteed that there would be at least one integer in the output.

Examples

standard input	standard output
`6 8`	`0`
`4 3 3 3 6 6`	`2`
`t`	`3 3 5 7 7`
`c 4`	
`a 5`	
`a 3`	
`r 5`	
`c 2`	
`t`	
`r 3`	

Note

Let us consider the changes in the database for the sample test:

1. $(4, 3, 3, 3, 6, 6)$
2. $(6, 6, 6, 3, 2, 2)$
3. $(6, 6, 6, 3, 2, 2, 5)$
4. $(6, 6, 6, 3, 2, 2, 5, 3)$
5. $(6, 6, 6, 3, 2, 2, 3)$
6. $(7, 7, 5, 3, 3, 3)$
7. $(7, 7, 5, 3, 3)$