Problem A. The Sum Again

Input:	*standard input*
Output:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

Implement a data structure that maintains a set S of integers, and supports the following operations:

- $add(i)$ — add an integer i to S (if it already contains this integer, the set doesn't change);
- $sum(l, r)$ — print the sum of all such elements x from S, that satisfy the inequality $l \le x \le r$.

Input

Initially S is empty. The first line of input contains n — the number of operations ($1 \le n \le 300\,000$). The following n lines contain operations. Each operation is either "`+` i", or "`?` l r". The operation "`?` l r" denotes $sum(l, r)$.

If the operation "`+` i" is the first one in the input, or if it follows another "`+`" operation, it denotes $add(i)$. If it follows "`?`" operation, and the result of that query is y, it denotes $add((i + y) \bmod 10^9)$.

Arguments of all operations are integers from 0 to 10^9, inclusive.

Output

For every sum query output one integer: the required sum.

Example

standard input	standard output
6 + 1 + 3 + 3 ? 2 4 + 1 ? 2 4	3 7

Problem B. K-th Maximum

Input:	*standard input*
Output:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

Implement a data structure that maintains a set S of integers, and supports the following operations:

- add an element to the set;
- remove an element to the set;
- return the element in the set that has the k-th maximal value in the set.

Input

The first line of input contains an integer n — the number of operations to perform ($1 \le n \le 100\,000$). The following n lines contain one operation description each. The operation is described with two integers c_i and k_i — the type of the opearation, and the argument of the operation.

The following operations must be supported:

- $+1$ or just 1: add an integer k_i to the set.
- 0: Find the k_i-th maximal value.
- -1: remove an integer k_i from the set.

It is guaranteed that when the element is added, it is not in the set yet, if it is removed it is in the set, and for the k-th maximum queries the set contains at least k elements.

Output

For every query operation print one line that contains the k_i-th maximum.

Example

standard input	standard output
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. Range Minimum Query

Input:	*standard input*
Output:	*standard output*
Time limit:	2 second
Memory limit:	256 mebibytes

Giggle company opens new offices in Singapre and Riga, and you are invited to the interview. Your task is to solve the following problem.

You have to create a data structure that would maintain a list of integer numbers. All elements of the list are numbered starting from 1. Initially the list is empty. The following two operations must be supported:

- query: “`? i j`” — return the minimal element between the i-th and the j-th element of the list, inclusive;
- modification: “`+ i x`” — add element x after the i-th element of the list. If $i = 0$, the element is added to the front of the list.

Of course, the performance of the data structure must be good enough.

Input

The first line of input contains n — the number of operations to perform ($1 \leq n \leq 200\,000$). The following n lines describe operations. You may consider that no boundary violations occur. The numbers stored in the data structure do not exceed 10^9 by their absolute value.

Output

For each query operation output its result on a line by itself.

Example

standard input	standard output
8 + 0 5 + 1 3 + 1 4 ? 1 2 + 0 2 ? 2 4 + 4 1 ? 3 5	4 3 1

Problem D. Move to Front

Input:	*standard input*
Output:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

Corporal Studip likes to give orders to his squad. His favorite order is "move to front". He lines the squad up in a line and gives a series of orders. Each order is: "Soldiers from l_i to r_i — move to front!"

Let us number the soldiers in the initial line up from 1 to n, from left to right. The order "soldiers from l_i to r_i — move to front!" makes the soldiers that are standing at the positions from l_i to r_i inclusive move to the beginning of the line, preserving their order.

For example, if at some moment the soldiers are standing in the following order: $2, 3, 6, 1, 5, 4$, after the order: "soldiers from 2 to 4 — move to front!" the order of soldiers is $3, 6, 1, 2, 5, 4$. If, for example, the order "soldiers from 3 to 4 — move to front!" follows, the new order of soldiers is $1, 2, 3, 6, 5, 4$.

Given the sequence of orders of corporal, find the final line up of the soldiers.

Input

The first line of the input file contains two integer numbers n and m — the number of soldiers and the number of orders ($2 \le n \le 100\,000$, $1 \le m \le 100\,000$). The following m lines contain orders, each line contains two integer numbers l_i and r_i ($1 \le l_i \le r_i \le n$).

Output

Output n integer numbers — the order of soldiers in the final line up, after executing all orders.

Example

standard input	standard output

6 3
2 4
3 5
2 2
	1 4 5 2 3 6

Problem E. Reverses

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

Physical education teacher of the high school arranges students in a row. Children like to fool the teacher, and sometimes "reverse" a segment of the row. That is, students at positions from l to r, inclusive, change their positions. The student at the position l swaps with the student at the position r, the student at the position $l+1$ swaps with the student at the position $r-1$, etc.

Help the teacher to answer queries about the sum of heights of students at some segment, while students sometimes reverse their order at some segments.

Input

The first line of input contains two integers n and m ($1 \le n, m \le 200\,000$) — the number of students and the number of queries to process. The second line contains n integers — the height of each student in order they are initially arranged in the row. The height of each student doesn't exceed 200 000.

The following m lines describe queries. Each query is specified using 3 integers q, l and r on a line ($0 \le q \le 1$, $1 \le l \le r \le n$). The value of q give the type of the query: 0 is the query to find the sum of heights of students at the positions from l to r, inclusive; 1 is the query to reverse the order of students at positions from l to r.

Output

For each query of type 0 output one integer, the answer to the query.

Example

standard input	standard output

5 6
1 2 3 4 5
0 1 5
0 2 4
1 2 4
0 1 3
0 4 5
0 3 5
	15
	9
	8
	7
	10

Problem F. Swapper

Input:	*standard input*
Output:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

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.

Example

standard input	standard output

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 G. Database

Input:	*standard input*
Output:	*standard output*
Time limit:	3 seconds
Memory limit:	256 mebibytes

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 ($1 \leq n, m \leq 200\,000$) — the number of students and the number of queries to perform, respectively.

The second line contains n integers g_i ($1 \leq g_i \leq 200\,000$) — 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 ($1 \leq x_i \leq 200\,000$) — 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
4 3 3 3 6 6
t
c 4
a 5
a 3
r 5
c 2
t
r 3
	0
	2
	3 3 5 7 7

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. $(7, 7, 5, 3, 3)$

Problem H. Permutations Strike Back

Input:	*standard input*
Output:	*standard output*
Time limit:	5 seconds
Memory limit:	256 mebibytes

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.

Example

standard input	standard output
4 2 1 2 3 4 GET 1 2 2 3 GET 1 3 1 3	1 3

Problem I. Key Insertion

Input:	*standard input*
Output:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

As an employee of the Macrohard Company, you have been asked to implement the new data structure that would be used to store some integer keys.

The keys must be stored in a special ordered collection that can be considered as an array A, which has an infinite number of locations, numbered starting from 1. Initially all locations are empty. The following operation must be supported by the collection: $Insert(L, K)$, where L is the location in the array and K is some positive integer value.

The operation must be processed as follows:

- If $A[L]$ is empty, set $A[L] \leftarrow K$.
- If $A[L]$ is not empty, perform $Insert(L + 1, A[L])$ and after that set $A[L] \leftarrow K$.

Given N integer numbers $L_1, L_2, \ldots, L_N$ you have to output the contents of the array after a sequence of the following operations:

$Insert(L_1, 1)$
$Insert(L_2, 2)$
$\ldots$
$Insert(L_N, N)$

Input

The first line of the input contains N — the number of $Insert$ operations and M — the maximal position that can be used in the $Insert$ operation ($1 \le N \le 131\,072$, $1 \le M \le 131\,072$).

The next line contains N integer numbers L_i that describe $Insert$ operations to be performed ($1 \le L_i \le M$).

Output

Output the contents of the array after a given sequence of $Insert$ operations. On the first line print W — the number of the greatest location that is not empty. After that output W integer numbers — $A[1], A[2], \ldots, A[W]$. Output zeroes for empty locations.

Example

standard input	standard output
5 4 3 3 4 1 3	6 4 0 5 2 3 1

Problem J. Amber Ball

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

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.

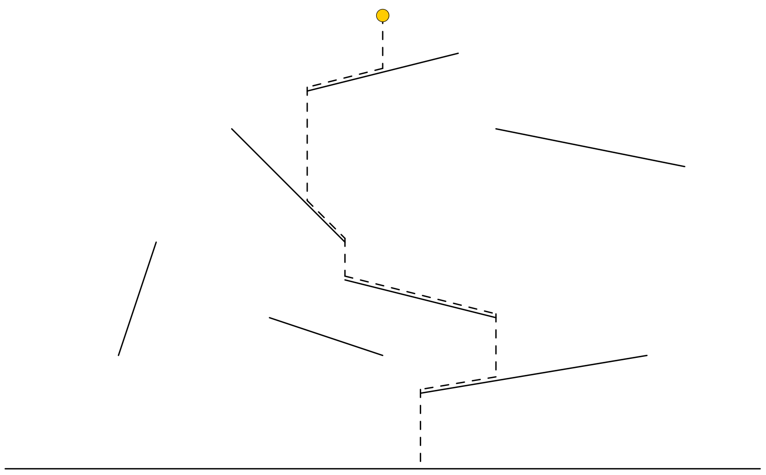

Figure 1: *
Third sample.

Input

The first line of the input contains a single integer n ($0 \le N \le 3 \cdot 10^5$) — 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 ($1 \le m \le 3 \cdot 10^5$) — 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

standard input	standard output

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.