Problem A. K-th Maximum

Input file:	`kthmax.in`
Output file:	`kthmax.out`
Time limit:	2 seconds
Memory limit:	64 megabytes

Your task is to implement the data structure that allows to add and remove elements and also to find the k-th maximum.

Input

The first line contains an integer n ($1 \le n \le 100\,000$) — the number of queries. Each of the following n line contains a description of the query. Every query is described by two integers c_i and k_i — the type and argument of query, respectively ($|k_i| \le 10^9$). The allowing commands are:

- $+1$ (or simply 1): add an element with the key k_i;
- 0: output the k-th maximum;
- -1: remove the element with the key k_i.

It is guaranteed that it does not need to keep elements with equal keys and remove nonexistent element. Also, it is guaranteed that the k_i-th maximum exists, when it is requested.

Output

For every command of type 0 output a single integer — the result of the query.

Example

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 B. Swapper

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

Before returning to the headquarters Aahz and Skeeve had to fill declaration in local customs. The result was a quite enormous sequence of numbers. It takes a lot of time to process this sequence.

— Swapper is bad, — customs official said knowledgeably.

— What is swapper? — curious Skeeve asked.

Aahz explained that swapper is data structure allowed the following queries

- in the range from x to y with even length ($y-x+1$ is divisile by 2) swap elements x with $x+1$, $x+2$ with $x+3$ and so on;
- calculate the sum in the arbitrary range.

Calculation can take a long time, so Myth corporation asks you to solve problem with swapper. Your task is to simulate queries effectively.

Input

The input file contains one or several tests. The first line of every test contains two integers n and m ($1 \le n, m \le 100\,000$) — the length of sequence and the number of queries. The second line of test contains n integers — the sequence. Every element of this sequence does not exeeced 10^6 by the absolute value.

Each of the following m lines contains a description of a query. A query of both kinds is defined with three integers: 1 x_i y_i for a query of the first kind, and 2 a_i b_i for a query of the second kind. Sum over all n and m in the input file does not exceed 200 000. The input file ends with the string of two zeros. It is guaranteed $x_i < y_i$ and $a_i < b_i$.

Output

For every test output responses to queries of the second type as shown in the example. The responses for different tests must be separated with empty line.

Example

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 C. Reverse

Input file:	`reverse.in`
Output file:	`reverse.out`
Time limit:	5 seconds
Memory limit:	256 megabytes

Physical education teacher has learned to count the total height of all students staying in a row on the positions from l to r a long time ago. But children play a trick on him. At some moment in time children on position from l to r are swapped. The teacher noticed that they always "reverse" the segment. That means l-th student swaps with r-th student, $(l+1)$-th student swaps with $(r-1)$-th student and so on. He decided to not to scold them, but to count the total height on all scheduled intervals.

Input

The first line contains two integers n and m ($1 \le n, m \le 200\,000$) — the number of children and the number of events. The second line contains n natural numbers — the heights of children in the order in the row. The height of every child does not exceed $2 \cdot 10^5$. Each of the following m line contains a description of an event: three numbers q, l, r ($0 \le q \le 1$, $1 \le l \le r \le n$). The number q shows a type of the event: 0 corresponds to a query to calculate the total height of children with numbers from l to r, 1 shows that children with numbers from l to r "reverse" their interval. All numbers in the input file are integers.

Output

For every event of kind 0 output a single number on the separate line — an answer for this query.

Example

reverse.in	reverse.out
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 D. Treap

Input file:	tree.in
Output file:	tree.out
Time limit:	1 seconds
Memory limit:	256 megabytes

Given pairs of integers (a_i, b_i) your task is to build a treap such that i-th node has keys (a_i, b_i). Nodes with keys a_i must satisfy binary search tree constraints and nodes with keys b_i must satisfy heap constraints.

Input

The first line contains single integer N ($1 \le N \le 50\,000$) — the number of pairs. The following N lines contain pairs (a_i, b_i). For all pairs $|a_i|, |b_i| \le 30\,000$ and $a_i \ne a_j$ for all $i \ne j$.

Output

If the treap with this set of keys does not exist, output `NO` in the first line. Otherwise, the first line must contain word `YES` and the following N lines must describe nodes of the treap. A node is described with three integers: number of the ancestor, number of the left child and number of the right child. If node does not have ancestor or some of child, output zero instead missing node.

Example

tree.in	tree.out
7 5 4 2 2 3 9 0 5 1 3 6 6 4 11	YES 2 3 6 0 5 1 1 0 7 5 0 0 2 4 0 1 0 0 3 0 0

Problem E. Sum

Input file:	sum2.in
Output file:	sum2.out
Time limit:	3 seconds
Memory limit:	256 megabytes

Consider a set S of integers. Your task is to implement the data structure that allows the following queries:

- $add(i)$ — add value i at the set S (if i is already in S, nothing happens);
- $sum(l, r)$ — output sum of all elements x from S satisfying the inequalities $l \le x \le r$.

Input

Initially the set S is empty. The first line contains an integer n ($1 \le n \le 300\,000$) — the number of queries. Each of the following n lines contains a description of the commands. Every line is either "$+\ i$" or "$?\ \ l\ r$". The line "? l r" issues query $sum(l, r)$. If line "$+\ i$" is either in the beginning of the input file or after another line starting with "$+$", this line issues query $add(i)$. Otherwise, if this line follows the line starting with "$?$" and the result of the previous query was y, it issues query $add((i + y) \bmod 10^9)$.

Arguments of every query belong to the interval from 0 to 10^9.

Output

For every line starting with "$?$" output a single number — the result of the corresponding query.

Example

sum2.in	sum2.out
6 + 1 + 3 + 3 ? 2 4 + 1 ? 2 4	3 7

Problem F. Move To Front!

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

Corporal Ducar enjoys to give commands to his troop. His favorite command is "Move to front!". Corporal aligns his soldiers in a row and gives some commands, and each of them is formulated as "Soldiers from l to r, move to front!". Before Ducar gave the first command, the soldiers had been numbered from 1 to n, from left to right. When soldiers hear a command "Soldiers from l to r, move to front!", soldiers with numbers from l to r move to the beginning of the row in the same order in which they are.

For example, if at some moment in time the soldiers stand in an order $1, 3, 6, 2, 5, 4$, then after the command "Soldiers from 2 to 3, move to front!" the new order is $3, 6, 1, 2, 5, 4$. If soldiers from 3 to 4 are moved to front by next command of corporal, than new order is already so $1, 2, 3, 6, 5, 4$.

Given the sequence of the commands of the corporal, your task is to find the order of soldiers after all commands.

Input

The first line contains two integers n and m ($1 \le n \le 100\,000$, $1 \le m \le 100\,000$) — the number of the soldiers and the number of commands, respectively. Each of the following m line contains description of the commands: two integers l_i and r_i ($1 \le l_i \le r_i \le n$).

Output

The single line must contain n integers — the order of the soldiers after the execution of all commands.

Example

movetofront.in	movetofront.out
6 3 2 4 3 5 2 2	1 4 5 2 3 6

Problem G. Key Values

Input file:	`key.in`
Output file:	`key.out`
Time limit:	2 seconds
Memory limit:	64 megabytes

You are employed by MacroHard company and as the first task they asked to implement new data structure. It should allow to keep integer key values.

This structure should be an array A of infinity length. Cells of this array are numbered starting from one. Initially all cells are empty. The only query that data stucture allows is *Insert(L, K)* where L is position in the array and K is some positive integer key value. The query perfoms in the following way:

- if cell $A[i]$ is empty, assign $A[L] := K$;
- otherwise, if cell $A[i]$ is not empty, execute *Insert*$(L + 1, A[L])$ and after execution assign $A[L] := K$.

Given the sequence of n integers $L_1, L_2, \ldots, L_n$ your task is to output the content of this array after execution the following queries:

Insert$(L_1, 1)$

Insert$(L_2, 2)$

...

Insert(L_N, N)

Input

The first line contains two integers N and M ($1 \le N \le 131\,072$, $1 \le M \le 131\,072$) — the number of queries and the maximal index that is allowed to use in *Insert* query. The following line contains n integers L_i — description of *Insert* queries ($1 \le L_i \le M$).

Output

Output the content of the array after execution of the given sequence of queries. The first line must contain integer W — index of the last non-empty cell. The second line must contain W integers — $A[1], A[2], \ldots, A[W]$. Output zero if cell is empty.

Example

key.in	key.out
5 4 3 3 4 1 3	6 4 0 5 2 3 1

Problem H. Implicit Key

Input file:	`implicitkey.in`
Output file:	`implicitkey.out`
Time limit:	1 seconds
Memory limit:	256 megabytes

Your task is to implement the data structure that allows to perfom the following queries on the array:

- *add i x* — add the element x after the i-th element of the array ($0 \le i \le len$);
- *del i* — remove the i-th element of the array.

Input

The first line contains two integers n and m ($1 \le n, m \le 10^5$) — the length of the initial array and the number of queries. The second line contains n integers — the elements of the initial array. All elements belong to the interval from 0 to $10^9 - 1$. The following m lines contain description of the queries as described above. It is guaranteed that all queries are correct. That means, for example, that i-th element exists, when it is requested to be removed.

Output

Output the final state of the array. The first line must contain the single integer — length of the array. The second line must contain the elements of the array separated by space.

Example

implicitkey.in	implicitkey.out
3 4 1 2 3 del 3 add 0 9 add 3 8 del 2	3 9 2 8

Problem I. Simple Binary Search Tree

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

Your task is to implement a binary search tree.

Input

The input file contains descriptions of queries performed on the tree. Number of queries does not exceed 100 000. Each line of the input file contains one of the following query:

- *insert x* — insert a key x into the tree;
- *delete x* — delete a key x from the tree;
- *exists x* — if there is key x in the tree, output "`true`", otherwise output "`false`";
- *next x* — output minimal element in the tree that strictly greater than x. If there is no such element, output "`none`";
- *next x* — output maximal element in the tree that strictly less than x. If there is no such element, output "`none`";

The integers in the all queries do not exeeced 10^9 by the absolute value.

Output

Output the results of the execution of all queries *exists*, *next* and *prev* from the input file. Follow the format of the output file in the example.

Example

bst.in	bst.out
insert 2 insert 5 insert 3 exists 2 exists 4 next 4 prev 4 delete 5 next 4 prev 4	true false 5 3 none 3

Problem J. Range Minimum Query

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

The Giggle company opens new office in France and you are invited for an interview. Your task is to implement the data structure that allows to perfom two queries on the array:

- $?\,i\,x$ — return the minimum element in the range from i to j;
- $+\,i\,x$ — insert element x after the i-th element of the array. If $i = 0$ then element is inserted at the beginning of the array.

Initially array is empty.

Input

The first line contains single integer n ($1 \le n \le 200\,000$) — the number of the queries.

The following n lines contain description of the queries as described above. It is guaranteed that all queries are correct. All values of the elements in the array do not exceed 10^9 by the absolute value.

Output

For every query of the first kind output its result on the separate line.

Example

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