Problem A. Balanced binary search tree

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

Implement balanced binary search tree.

Input

The input contains several operations with tree, their amount doesn't exceed 10^5. Every line contains one of the following operations:

- `insert` x — insert key x into tree. If there's a key x in the tree already, do nothing;
- `delete` x — remove key x from tree. If there's no key x in the tree, do nothing;
- `exists` x — if there's a key x in the tree, output "`true`", otherwise output "`false`";
- `next` x — output smallest key in the tree, which is strictly larger than x, or "`none`" if there's no such key;
- `prev` x — output maximum key in the tree, which is strictly less than x, or "`none`" if there's no such key.

All keys are integers no greater than 10^9 by absolute value.

Output

Output results of all operations `exists`, `next`, `prev`.

Example

standard input	standard output
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 B. sum2

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

Implement a data structure that supports the set of S integers with which the following operations are allowed:

- add(i) — add the number i to the set S (if it is already there, the set does not change);
- sum(l, r) — output the sum of all elements x from S that satisfy the inequality $l \le x \le r$.

Input

Initially, the set S is empty. The first line of the input file contains n — the number of operations ($1 \le n \le 300\,000$). The next n lines contain operations. Each operation has the form either « `+` i » or « `?` l r ». Operation « `?` l r » sets the query to sum(l, r).

If the operation « `+` i » is in the input file at the beginning or after another operation « `+` », then it defines the operation add(i). If it goes after the query « `?` », and the result of this query was y, then the operation add($(i + y) \bmod 10^9$) is performed.

In all queries and adding operations, the parameters are in the range from 0 to 10^9.

Output

For each request print one number — response to the request.

Example

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

Problem C. Move to front

Input file:	`standard input`
Output file:	`standard output`
Time limit:	6 seconds
Memory limit:	512 megabytes

You have an array $a_1 = 1, a_2 = 2, \ldots, a_n = n$ and a sequence of queries: move elements from l_i to r_i to front.

For example, if the array is $2, 3, 6, 1, 5, 4$, after the query $(2, 4)$ the new order of elements in the array is $3, 6, 1, 2, 5, 4$.

If, for example, the query $(3, 4)$ follows, the new order of elements is $1, 2, 3, 6, 5, 4$.

Print the final order of elements in the array.

Input

The first line of the input file contains two integer numbers n and m ($2 \le n \le 100\,000$, $1 \le m \le 100\,000$) — the number of elements and the number of queries. The following m lines contain queries, 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 elements in the final array, after executing all queries.

Example

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

Problem D. Reverses

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

You are given an array $a_1 = 1, a_2 = 2, \ldots, a_n = n$ and a sequence of operations: reverse order of elements from position l_i to position r_i.

For example, for array $1, 2, 3, 4, 5$, after operation $(2, 4)$, new order will be $1, 4, 3, 2, 5$. And then after applying $(3, 5)$ new order will be $1, 4, 5, 2, 3$.

Output order of elements after all operations.

Input

First line contains integers n and m ($2 \le n \le 100\,000$, $1 \le m \le 100\,000$) — amount of elements in the array and number of operations. Next m lines contains description of operations. Every line contains two integers l_i and r_i ($1 \le l_i \le r_i \le n$).

Output

Output n integer numbers — the order of elements in the final array, after executing all operations.

Example

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

Problem E. Log Analysis

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

Lisa writes a log analysis application for a distributed computer system. Unlike single-node logs, that are append-only the distributed log is highly volatile. When a node becomes online, it may push a batch of events in the past of the log. Conversely, when it goes offline some log entries may disappear.

To ensure the stability and availability of the application Lisa need to monitor the number of distinct events in the log segments. She is going to handle distributed part, while you have to implement a local one.

Your program is started from the empty log and must support the following operations:

- **insert** $\langle index \rangle$ $\langle number \rangle$ $\langle type \rangle$ — inserts $\langle number \rangle$ of events of type $\langle type \rangle$ before event with index $\langle index \rangle$. Events with indices larger or equal to $\langle index \rangle$ are renumbered.
- **remove** $\langle index \rangle$ $\langle number \rangle$ — removes $\langle number \rangle$ of events starting from event with index $\langle index \rangle$.
- **query** $\langle index_1 \rangle$ $\langle index_2 \rangle$ — counts the number of distinct event types for events with indices from $\langle index_1 \rangle$ to $\langle index_2 \rangle$ inclusive.

The events are indexed starting from 1. The event types are represented by single-letter codes.

Input

The first line of the input file contains single integer number n — the number of operations ($1 \le n \le 30\,000$). The following n lines contain one operation description each.

Operation description starts with operation type: '`+`' for insert, '`-`' for remove and '`?`' for query. Operation type is followed by operation arguments.

All indices are valid, i. e. events with specified indices exist, and you never have to remove events past the end of the log.

The $\langle number \rangle$ for the insert and remove operations does not exceed 10 000.

Event types are represented by lowercase Latin letter.

Output

For each query operation output a single number — the number of distinct event types between $\langle index_1 \rangle$ and $\langle index_2 \rangle$ inclusive.

Example

standard input	standard output
8 + 1 4 w + 3 3 o ? 2 3 - 2 2 ? 2 3 + 2 2 t ? 1 6 - 1 6	2 1 3

Problem F. Key insertion

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

You are working in the company Macrohard. And you are asked to implement a data structure that will store set of integer keys.

Suppose that keys are stored in infinity array A, indexed from 1. Initially, all its elements are empty. The structure should have the following operation:

`Insert(`L`, `K`)`, where L is a position in the array, and K is some positive integer.

The operation should work as follows:

- If element $A[L]$ is empty, then assign $A[L] \leftarrow K$.
- Otherwise $A[L]$ isn't empty, call `Insert(`$L+1$`, `$A[L]$`)` and then assign $A[L] \leftarrow K$.

By given N integers $L_1, L_2, \ldots, L_N$ output an array after calling operations:

`Insert(`L_1`, 1) Insert(`L_2`, 2) ... Insert(`L_N`, `N`)`

Input

First line contains integers N and M — number of `Insert` operations and maximum position which is used in one of insert operations `Insert` ($1 \le N \le 131\,072$, $1 \le M \le 131\,072$).

Next line contains N integers L_i, which describes position of `Insert` operations, which you should call ($1 \le L_i \le M$).

Output

Output an array after calling all operations. First line should contain integer W — largest index of a non-empty element in the array. Then output W integers — $A[1], A[2], \ldots, A[W]$. Output zeroes for empty elements.

Example

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