# Binary Search Tree Test

**Time limit:** 2.5s **Memory limit:** 1G
Java: 4.5s
Racket: 4.5s

**Xyene** is doing a contest. He comes across the following problem:

You have an array of $N$ $(1 \leq N \leq 100\,000)$ elements. There are $M$ $(1 \leq M \leq 500\,000)$ operations you need to perform on it.

Each operation is one of the following:

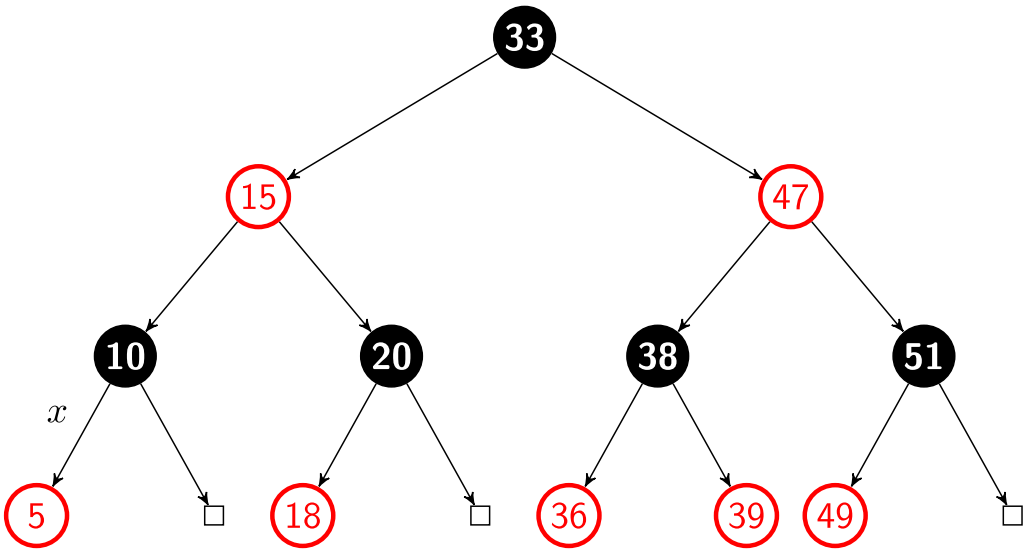
- `I v` Insert $v$ into the array.
- `R v` Remove a single element from the array with a value of $v$, if it exists.
- `S v` Output the $v^{th}$ smallest element in the array. It is guaranteed that $v$ does not exceed the size of the array.
- `L v` Output the index, starting from $1$, of the first occurrence of $v$ in the array, if the array was sorted. Output $-1$ if $v$ is not present in the array.

After all of the operations, print out all of the elements remaining in the array in non-decreasing order.

**To enforce performing operations in an online manner, $v$ will be encrypted.**

At any time, every element in the array is between $1$ and $10^9$, inclusive.

**Xyene** knows that one fast solution uses a Binary Search Tree. However, he knows that a standard binary search tree has a worst case runtime of $\mathcal{O}(N)$ per operation. He practices different data structures every day, but still somehow manages to get them wrong. Will you show him a working example?




*Not a binary search tree.*

## Input Specification

The first line has $N$ and $M$.

The second line has $N$ integers, the original array.

The next $M$ lines each contain an operation in the format `C x`, where $C$ is the type of operation. $v$ is encrypted: you should decode it by finding $v = x \oplus lastAns$, where $lastAns$ is the answer to the previous `S` or `L` operation (or $0$ if neither operation has occurred). You should perform the operation using $v$. $\oplus$ denotes the bitwise XOR operation.

## Output Specification

For each `S` or `L` operation, output the answer on its own line.

After all operations have been finished, output all of the elements in the final array in non-decreasing order on a single line.

## Sample Input

```
5 8
9 4 8 11 2
S 4
I 1
S 13
R 10
L 10
L -5
I 8
L 8
```

## Sample Input (Not Encrypted)

For your convenience, here is a version of the sample input that is **NOT** encrypted. Remember, all of the real test files will be encrypted (like the input above).

```
5 8
9 4 8 11 2
S 4
I 8
S 4
R 2
L 2
L 4
I 9
L 9
```

## Sample Output

```
9
8
-1
1
4
4 8 8 9 9 11
```