Lucky Draw

Statement

Mr. Panda has been tasked with hosting a lucky draw for a company with Q people. However, he finds normal ways of picking the winner too boring. Thus, he has come up with a very peculiar way of choosing the winner, he will use a bag of numbered balls.

He starts off with a bag containing N numbered balls and passes it around the employees. Each employee can either **add a ball**, with a specific number of that employee's choosing, into the bag, or **remove a ball** from the bag. However, there should be **at least 2 balls in the bag at all times**. These actions will be represented by Q actions of the following format:

- `ADD [K]` – Add one ball with number K into the bag.
- `REMOVE [K]` – Remove one ball with number K from the bag. It is guaranteed there is a ball with number K in the bag.

At the end, Mr. Panda will ask each employee to pick out two separate balls from the bag. The **absolute difference** between the numbers on the balls will be their lottery number and the lowest number among all the employees wins! As Mr. Panda wants to save time, he wants to know what the **lowest possible winning number** is because if any employee gets that, he can immediately declare that employee the winner without going through the rest of the employees! Do note that the two *different* balls chosen can have the same value, as long as they are two **separate balls**.

However, he is not going to look through so many balls inside the bag only at the end, so he wants to keep track of the **lowest possible winning number** after every action that an employee performs. As Mr. Panda is too busy preparing the balls and the bag, he asks you, a competent CS2040 programmer, to help him code a program that can help him keep track of this. Meanwhile, Rar the Cat walks by and hints that the method `Math.abs` might be useful for your implementation.

Constraints

The following constraints apply for all subtasks.

- $2 \leq N \leq 10^5$
- $1 \leq Q \leq 10^5$
- $-10^8 \leq K \leq 10^8$

Further subtask-specific constraints will be listed below, under each subtask.

Input

The first line of input will contain an integer, N. This denotes the number of balls in the bag initially.

The next N lines will contain an integer each, representing the number of the balls currently in the bag.

The next line of input will contain an integer Q, the number of actions. Each action will have one of the following input formats:

- `ADD [K]` – Add one ball with number K into the bag.
- `REMOVE [K]` – Remove one ball with number K from the bag. It is guaranteed there is a ball with number K in the bag.

Output

For every action, output the smallest difference of numbers between any pair of balls inside the bag.

Examples

Sample Input	Expected Output
3 4 7 2 4 ADD 6 REMOVE 7 REMOVE 4 ADD 2	1 2 4 0

After the first action, the smallest possible difference is that between 6 and 7 which is 1.

After the second action, the smallest possible difference is that between 2 and 4 which is 2.

After the third action, the smallest possible difference is that between 2 and 6 which is 4.

After the fourth action, the smallest possible difference is that between 2 and 2 which is 0. Take note that this only happens in the last case since there are now 2 distinct balls in the bag with number 2.

Subtasks

You may choose to solve any one of the following subtasks.
This accounts for 70% of your score for this problem.

- **Subtask 1 (20%)**:
  To obtain full credit for this subtask, your solution should run in worse than $O(QN \log N)$ time.
- **Subtask 2 (40%)**:
  To obtain full credit for this subtask, your solution should run in $O(QN \log N)$ time or better.
- **Subtask 3 (60%)**:
  To obtain full credit for this subtask, your solution should run in $O(QN)$ or $O(K_{\max} + Q + N)$ time or better.
- **Subtask 4 (80%)**:
  To obtain full credit for this subtask, your solution should run in $O((Q+N)\sqrt{Q+N})$ time or better.
- **Subtask 5 (100%)**:
  To obtain full credit for this subtask, your solution should run in $O((Q+N)\log(Q+N))$ time or better.

Marking Scheme

- Correctness and Efficiency (70%)
  - Refer to the Subtasks section.
- Programming Style (30%)
  - Purpose of methods and statements (5%)
  - Pre- and post- conditions (5%)
  - Modularity (5%)
  - Scoping (5%)
  - Meaningful Identifiers (4%)
  - Variable Naming (1%)
  - Identation (5%)

Notes

1. A skeleton file has been given to help you. You should not create a new file or rename the file provided. You should develop your program using this skeleton file.
2. You are free to define your own helper methods and classes (or remove existing ones) if it is suitable but you must put all the new classes, if any, in the same skeleton file provided.

Skeleton File

You are given the skeleton file `LuckyDraw.java`. You should see the following contents when you open the file:

```java
/**
 * Name          :
 * Matric. No    :
*/

import java.util.*;

public class LuckyDraw {
    private void run() {
        // implement your "main" method here
    }

    public static void main(String args[]) {
        LuckyDraw runner = new LuckyDraw();
        runner.run();
    }
}
```