Boxes

Cats like to sit in cardboard boxes. Hence, Rar the Cat and his friends bought a total of **R*C** identical cardboard boxes and arranged them in a grid formation of **R** rows and **C** columns. For ease of reference, the rows are numbered from **0** to **R-1** from top to bottom and the columns are numbered from **0** to **C-1** from left to right. Each box also has a coordinate equals to the row number and the column number of the box. For example, given below is how the boxes will look like if **R** = 2 and **C** = 5. The values in the boxes are the coordinates of the box.

(0, 0)	(0, 1)	(0, 2)	(0, 3)	(0, 4)
(1, 0)	(1, 1)	(1, 2)	(1, 3)	(1, 4)

Cats do not like to share boxes. Some very picky cats don't even like to share the entire row or column of boxes with other cats. As the number of boxes increases, he has difficulty managing the usage of these boxes such that no two cats will accidentally end up sitting in the same box. Hence, he envisions to have a system that supports the below operations to help him:

Operation	Description
`SIT [x] [y]`	Records that a cat is now sitting in the box with coordinate (**[x]**, **[y]**). It is guaranteed there is no existing cat in the box with coordinate (**[x]**, **[y]**).
`BOX [x] [y]`	Checks if there is a cat sitting in the box with coordinate (**[x]**, **[y]**). If there is a cat, output 'Y'. Otherwise, output 'N'.
`ROW [x]`	Checks if there is a cat sitting in any of the boxes in *row* **[x]**. If there is at least one cat sitting in row **[x]**, output 'Y'. Otherwise, output 'N'.
`COL [y]`	Checks if there is a cat sitting in any of the boxes in *column* **[y]**. If there is at least one cat sitting in column **[y]**, output 'Y', Otherwise, output 'N'.

You may assume that the system starts off with **having no cats in all boxes.**

Input

The first line of input will contain integers **Q, R** and **C**, the number of operations to be performed on the system, the number of rows of boxes and the number of columns of boxes respectively.
Q lines will follow, representing an operation each. The operations should be executed in order and the format would be as described in the table above. (See sample)

Output

You do not have to output anything for the **SIT** command.
For **BOX** command, output 'Y' if there is a cat in the box with coordinate (**[x]**, **[y]**) and 'N' if there isn't.
For **ROW** command, output 'Y' if there is a cat in any one of the boxes in row **[x]** and 'N' if there isn't.
For **COL** command, output 'Y' if there is a cat in any one of the boxes in column **[y]** and 'N' if there isn't.

Limits

- $1 \le \mathbf{Q} \le 100{,}000$
- $1 \le \mathbf{R}, \mathbf{C} \le 10^9$
- All **[x]** parameters will be between $0 \le x < \mathbf{R}$ and all **[y]** parameters will be between $0 \le y < \mathbf{C}$.

Sample Input (**boxes1.in**)	Sample Output (**boxes1.out**)
11 2 2 SIT 0 1 BOX 0 1 BOX 0 0 BOX 1 1 ROW 0 ROW 1 COL 0 COL 1 SIT 1 1 ROW 1 COL 1	Y N N Y N N Y Y Y
Sample Input (**boxes2.in**)	Sample Output (**boxes2.out**)
5 1000 1024 SIT 999 1023 SIT 950 1000 ROW 999 COL 1000 BOX 999 1000	Y Y N

Notes:

1. You should develop your program in the subdirectory **ex3** and use the skeleton java file provided. You should not create a new file or rename the file provided.
2. You are free to define your own helper methods and classes (or remove existing ones).
3. Please be reminded that the marking scheme is:
   a. Public Test Cases (1%) - 1% for passing **all** test cases, 0% otherwise
   b. Hidden Test Cases (1%) - Partial scoring depending on test cases passed
   c. Manual Grading (1%)
      i. Overall Correctness (correctness of algorithm, severity of bugs)
      ii. Coding Style (meaningful comments, modularity, proper indentation, meaningful method and variable names)
4. Your program will be tested with a time limit of not less than **2 sec** on Codecrunch.

Skeleton File – Boxes.java

You are given the below skeleton file `Boxes.java`. You should see a non-empty file when you open the skeleton file. Otherwise, you might be in the wrong working directory.

```
/**
 * Name         :
 * Matric. No   :
 * PLab Acct.   :
 */

import java.util.*;

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

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