Shooting

Rar the Cat is so engrossed in playing a Role-Playing Game (RPG) that he has no time to set questions for the Practical Exam! This RPG consists of an **N** x **N** grid. Rows are labelled 0 to **N** - 1 from top to bottom, and columns are labelled 0 to **N** - 1 from left to right. The square in the **r**-th row and **c**-th column is denoted by the coordinates (**r, c**). Rar the Cat starts at the top left corner, denoted by coordinates (0, 0). There are initially no enemies on the grid, but enemies spawn one by one during the game.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 |
| 0 | C:\Users\Si Jie\Desktop\cat.jpg |  |  |  |  |  |
| 1 |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |
| 5 |  |  |  |  |  |  |

**Figure 1**: There are 5 enemies on this 6x6 grid: [(1, 2), (1, 5), (2, 0), (3, 3), (5, 3)].

Rar the Cat is at (0, 0)

You need to simulate the game by supporting the following operations:

|  |  |
| --- | --- |
| **Operation** | **Description** |
| SHOOT **[direction]** | **[direction]** will only be **UP**, **DOWN**, **LEFT**, or **RIGHT**.  Rar the Cat shoots a bullet in the corresponding direction from where he is. The closest enemy in that direction will be hit and is removed from the game. If there are no enemies in that direction, the bullet misses.  If there are no enemies in that direction, **output “MISSED”**. Otherwise, **output the coordinate of the enemy hit**. |
| TELEPORT **[r] [c]** | Rar the Cat will move to a new coordinate (**[r], [c]**).  It is guaranteed that (**[r], [c]**) will be a valid cell on the grid and contains no enemies.  It is possible that Rar the Cat does not move as (**[r]**, **[c]**) could be his current coordinates. |
| SPAWN **[r] [c]** | Adds a new enemy at the coordinate (**[r], [c]**).  It is guaranteed that (**[r], [c])** will be a valid cell on the grid that is empty. |

At the end of all the operations, Rar the Cat also wants to know all the enemies that are remaining (i.e. have not been shot). Thus, you should also output all remaining enemies sorted in **increasing** row number, breaking ties by **increasing** column number. For each coordinate, print the row number then the column number with one space in between.

**Input**

The first line of input contains 2 integers **N** and **Q**. **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**

For every **SHOOT** operation**,** if the bullet hits an enemy, output the coordinates of the enemy with the row number first followed by the column number. If the bullet misses, output “**MISSED**”.

At the end of all **Q** operations, output the coordinates of all enemies in **increasing** row number, breaking ties by **increasing** column number. **Do not print a space after the last coordinate.** Instead, remember to print an end-line character at the end of the output.

**Limits**

* 1 ≤ **N** ≤ 150,000
* 1 ≤ **Q** ≤ 500,000
* It is guaranteed that all the coordinates given will be a valid cell on the grid i.e. 0≤ **[r], [c]** < N.
* It is guaranteed that for a **TELEPORT** operation, the cell will not contain an enemy.
* It is guaranteed that for a **SPAWN** operation, the cell will not contain Rar the Cat or another enemy that is still not shot yet.
* It is guaranteed that there is at least one enemy at the end of all the operations.

|  |  |
| --- | --- |
| Sample Input (**shooting1.in**) | Sample Output (**shooting1.out**) |
| 6 12  SPAWN 1 2  SPAWN 1 5  SPAWN 2 0  SPAWN 3 3  SPAWN 5 3  SHOOT DOWN  SHOOT DOWN  TELEPORT 1 0  SHOOT RIGHT  SHOOT RIGHT  SPAWN 3 5  SPAWN 5 5 | 2 0  MISSED  1 2  1 5  3 3 3 5 5 3 5 5 |

**Explanation**

After the first 5 operations, the game screen looks like Figure 1. Rar the Cat then shoots downwards twice in the 6th and 7th operations. The first bullet hits the enemy at (2, 0). The second bullet misses.

Rar the cat then moves to (1, 0), where there are 2 enemies on the same row. He proceeds to shoot both of them. Notice that bullets do not pass through enemies.

Finally, two new enemies spawn at (3, 5) and (5, 5), and the game ends with 4 enemies remaining: [ (3, 3), (3, 5), (5, 3), (5, 5)]. The enemies should be printed in ascending order of **r**, then by **c**. Hence, the last line of the output is 3 3 3 5 5 3 5 5. Note that there should be no additional space after the last number.

**Notes:**

1. You should develop your program in the subdirectory **ex4** 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:
   1. Public Test Cases (1%) - 1% for passing **all** test cases, 0% otherwise
   2. Hidden Test Cases (1%) - Partial scoring depending on test cases passed
   3. Manual Grading (1%)
      1. Overall Correctness (correctness of algorithm, severity of bugs)
      2. 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.