Transmission

Rar the Cat is building a new system, dubbed the Cats Transmission System (CTS). This horribly inefficient system utilises cats to help transfer information. In this system, there are **N** cats lined up in a line from cat **0** to cat **N-1**. If a message needs to be transferred from, say, cat 2 to cat 7, cat 2 will pass on the message to cat 3, to cat 4... and so on, until it reaches cat 7. Sounds simple, right?

However, there is a problem. As everyone knows, cats *LOVE* sleeping. Some of these cats tend to fall asleep on the job. Say, if cat 3 falls asleep, the message from cat 2 to cat 7 will not be able to be transmitted. As such, given a list of "SLEEP" and "WAKE" events, as well as "TRANSMIT" requests in between, Rar wants you to check if each of these "TRANSMIT" requests will pass. All cats start out awake.

There will be a total of **Q** events. The format of the events will be as follows:

Event	**Description**
`WAKE` **`[x]`**	Cat **[x]** wakes up
`SLEEP` **`[x]`**	Cat **[x]** falls sleep
`TRANSMIT` **`[x] [y]`**	Attempt to transmit information from cat **[x]** to cat **[y]** (**[x]** ≤ **[y]**). If it is successful (all cats from **[x]** to **[y]** inclusive are awake), then output “YES”. Otherwise, output “NO”.

Input

The first line of input will contain two integers, **N** and **Q**.
The next **Q** lines of input will each contain one event as stated above.

Output

There should be one line of output for every "TRANSMIT" operation, either stating a "YES" or a "NO".

Limits

- $0 < \mathbf{N} \le 2^{31}$-1
- $0 < \mathbf{Q} \le 300{,}000$.
- All values of **x** and **y** are guaranteed to be between **0** and **N-1**.

Sample Testcase

Sample Input (**transmission1.in**)	Sample Output (**transmission1.out**)

`8 8`
`TRANSMIT 2 7`
`SLEEP 6`
`TRANSMIT 1 7`
`TRANSMIT 1 5`
`SLEEP 4`
`TRANSMIT 1 3`
`WAKE 4`
`TRANSMIT 1 5`
	`YES`
	`NO`
	`YES`
	`YES`
	`YES`

Explanation

There is a total of 8 cats, labelled from 0 to 7, and 8 events that follows.

Initially, all the 8 cats are awake. Hence, the transmission request from cats 2 to 7 will succeed.

Then, cat 6 falls asleep. Subsequently, the transmission request from cats 1 to 7 will fail as cat 5 cannot transfer the information to cat 6, which is sleeping.

However, the transmission request from cats 1 to 5 will still succeed as every cat in the range is still awake.

Then, cat 4 falls asleep. The next transmission request from cats 1 to 3 still succeeds as every cat in the range is still awake.

Cat 4 now wakes up. The transmission request from cats 1 to 5 will succeed as every cat in the range is now awake.

Notes:

1. You should develop your program in the subdirectory **ex2** 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 – Transmission.java

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

You should see the following contents when you open the skeleton file:

```java
import java.util.*;
public class Transmission {
    private void run() {
        //implement your "main" method here
    }
    public static void main(String[] args) {
        Transmission newTransmission = new Transmission();
        newTransmission.run();
    }
}
```

Source

CS2040 AY2017/18 Semester 2 Take Home Lab 4