Speakers

Mr. Panda is organising an event where he invites guest speakers to talk. Each speaker is invited to talk for **L** time units. This duration is fixed for all speakers. This means that if any speaker starts their speech at time **S,** the speech will end at time (**S + L – 1**). However, due to their busy schedules, each speaker can only come for a specific time period of **L** time units and some of these might clash since only 1 speaker can be talking at any time during the event. For example, if there is already a speaker who starts their speech at time **S**, then another invited speaker must either **end their speech before time S** or **start their speech from time S + L** onwards.

In order to make things easy for himself, Mr. Panda decides to schedule the speakers on a first come first serve basis. This means that when a speaker tells Mr. Panda their schedule, Mr. Panda will only allow the speaker to speak if their speech does not clash with that of any other speaker that is currently scheduled. Otherwise, he will simply reject the speaker.

Furthermore, some speakers might decide to pull out halfway. If this is done, Mr. Panda does not re-consider the speakers that he has rejected since this will look bad on him. In summary, Mr. Panda needs to implement a **Data Structure** that can support the following operations:

Operation	Description
`INSERT [S]`	Attempt to insert a speech that starts at time **S** and ends at (**S + L – 1)**. If there is no other speech that clashes with speech, output **"Y"** on one line and add it to the schedule. Otherwise, output **"N"** on one line and ignore the operation.
`REMOVE [S]`	Attempt to remove a speech that starts at exactly time **S** and ends at (**S + L – 1)**. If such a speech exists, output **"Y"** and remove it from the schedule. Otherwise, output **"N"** on one line and ignore the operation.

Lastly, Mr. Panda wants you to list the final schedule by listing all the start times of the speeches in **increasing** order. It is guaranteed that there is at least one speech in the schedule at the end of all the operations.

Input
The first line of input contains two integers **Q** an **L**. **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 **INSERT** and **REMOVE** operation, output **"Y"** or **"N"** according to the description.
At the end of all **Q** operations, output all the starts times of the speeches in the schedule in **increasing** order. Add a single space between two consecutive start times. **Do not print a space after the last start time.** Instead, remember to print an end-line character at the end of the output.

Limits
- $1 \le \mathbf{Q} \le 200{,}000$
- $1 \le \mathbf{L} \le 10^{18}$
- All the start times will range from 1 to 10^{18} inclusive.
- Make sure to use the **'long'** 64-bit data type to store the value of **L** and the start times.
- It is guaranteed that there is at least one speech in the schedule at the end of all the operations.

Sample Input (**speakers1.in**)	Sample Output (**speakers1.out**)
10 4 INSERT 5 INSERT 2 INSERT 8 INSERT 1 INSERT 9 REMOVE 2 REMOVE 9 INSERT 12 INSERT 10 REMOVE 5	Y N N Y Y N Y Y N Y 1 12

Explanation

Operation	Result	Schedule
INSERT 5	Successfully adds [5, 8]	[5, 8]
INSERT 2	[2, 5] clashes with [5, 8]	[5, 8]
INSERT 8	[8, 11] clashes with [5, 8]	[5, 8]
INSERT 1	Successfully adds [1, 4]	[1, 4], [5, 8]
INSERT 9	Successfully adds [9, 12]	[1, 4], [5, 8], [9, 12]
REMOVE 2	No such speech in schedule.	[1, 4], [5, 8], [9, 12]
REMOVE 9	Successfully removes [9, 12]	[1, 4], [5, 8]
INSERT 12	Successfully adds [12, 15]	[1, 4], [5, 8], [12, 15]
INSERT 10	[10, 13] clashes with [12, 15]	[1, 4], [5, 8], [12, 15]
REMOVE 5	Successfully removes [5, 8]	[1, 4], [12, 15]

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 – Speakers.java

You are given the below skeleton file `Speakers.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 Speakers {
    private void run() {
        //implement your "main" method here
    }
    public static void main(String[] args) {
        Speakers newSpeakers = new Speakers();
        newSpeakers.run();
    }
}
```