

Customs

Mr. Pigcow is the manager of the customs between Singapore and Yishun (which has recently seceded). A lot of Singaporeans want to get into Yishun to view the popular Point in the North so people are continuously joining the back of the queue. However, there are about 98,696 application forms to check through, so the customs officer can only process a new visitor at the rate of one a year, and as a result the queue is horrifyingly long.

Mr. Pigcow is worried that the customs officer is too ugly as they keep observing people at the back of the queue leaving the queue. They want to find out whether the looks of the customs officer is the cause of this problem. Thus, they want to know at any given time what is the number of people that can see the customs officer. A person can see the customs officer if everyone in the queue in front of him/her is shorter than him/her, or if he/she is the first person in the queue.

Not wanting to bother the existing customs officer from his work, Mr. Pigcow has asked you, a programmer with nothing better to do, to develop an algorithm to help him solve his problem.

Input

The first line of the input consists of a single integer, Q ($1 \leq Q \leq 100000$), the number of queries you need to answer. It is then followed by Q lines, each containing a single query for you to answer. The queries follow the following specification:

Query Type Input Format: **<QUERY_TYPE> <APPROPRIATE_PARAMETERS>**

1. **join HEIGHT**
A deluded person of height HEIGHT joins the back of the queue, in the hopes that he will be able to enter Yishun by the end of the century. Print the total number of people that can see the customs officer after this person joins the queue.
2. **leave PEOPLE**
Finally realizing that entering Yishun is practically impossible, the last PEOPLE people from the back of the queue leave. It is guaranteed that PEOPLE will be at least 1 and at most the current size of the queue. Print the total number of people that can see the customs officer after these people leave the queue.

Output

Print the result of all the queries as described in the input format above. The last line of the output must contain a newline character. In the sample output below, the first line is left empty for better clarity of the sample. No blank lines are to be printed in the actual output.

Sample Input

```

12
join 200
leave 1
join 160
join 150
join 160
join 170
join 180
leave 1
join 165
leave 3
join 165
leave 3

```

Sample Output

```

1
0
1
1
1
2
3
2
2
1
2
0

```

Explanation

The queue after each of the queries is shown below. The cells highlighted in green are the people who can see. The person with height 160 at nearer to the back is blocked by the person with height 160 at the front.

join 200	200				
leave 1					
join 160	160				
join 150	160	150			
join 160	160	150	160		
join 170	160	150	160	170	
join 180	160	150	160	170	180
leave 1	160	150	160	170	
join 165	160	150	160	170	165
leave 3	160	150			
join 165	160	150	165		
leave 3					

Skeleton

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

Notes:

1. You should develop your program in the subdirectory **ex1** and use the skeleton java file provided. You should not create a new file or rename the file provided.
2. If your algorithm is different from the given skeleton, you are free to write a solution according to your own algorithm. However, **your algorithm must use either stack or queue** to solve this problem, whichever is suitable. You are not allowed to use arrays, ArrayList, HashMap, etc. for this problem.
3. You are **free to define your own classes (or remove existing ones)** if it is suitable.
4. Please be reminded that the marking scheme is:

Input : 10%

Output : 10%

Correctness : 50%

Programming Style : 30% (awarded if you score **at least 20% from the above**):

- o Meaningful comments (pre- and post- conditions, comments inside the code): 10%
- o Modularity (modular programming, proper modifiers [public / private]): 10%
- o Proper Indentation: 5%
- o Meaningful Identifiers (for both method and variable names): 5%

Compilation Error : Deduction of **50% of the total marks obtained**.

Skeleton File – Customs.java

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

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

import java.util.*;

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

    public static void main(String[] args) {
        ClassPhoto newCustoms = new Customs();
        newCustoms.run();
    }
}
```