

## Ball Passing

Mr. Panda is an orientation group leader and sets up a simple ice breaking game. **N** students initially sit in a circle with one of them holding a ball. Every time Mr. Panda shouts “NEXT”, the student holding the ball passes it to the next student clockwise. The student that receives the ball will have to shout their name.

Occasionally, the student holding the ball might want to “LEAVE” the circle (going to the toilet for example). In that case, the student will pass the ball to the next student clockwise in the circle and leave. The student receiving the ball will shout their name as well.

New students might also arrive and want to “JOIN” the circle. In this case, the new student will sit at the position such that they will be the next student to receive the ball. The ball will then be immediately passed to the new student and the new student will have shout their name.

As the number of students increases, Mr. Panda finds it very difficult to keep track of all the students and their names. Thus, given the list of events that happen (either NEXT, LEAVE or JOIN), he wants you to code a program to tell him what name will be shouted at each time.

Even though the students are sitting in a circle which is non-linear, Rar the Cat suggests that clever usage of Java API **Linear** Data Structures would be sufficient to solve this task. (i.e. You do not need to implement your own data structure for this task.)

### Input

The first line contains a single integer **N**, the number of students initially in the circle.

The second line contains **N** strings, representing the names of the students in the circle in a clockwise manner. The first name in the list is the name of the student currently holding the ball.

The third line contains a single integer **Q**, the number of events that happen.

The next **Q** lines represent the events that happen in chronological order. The first string of each line will either be NEXT, LEAVE or JOIN, representing the event. If the event is JOIN, then that line will contain a second string representing the name of the student that joined.

### Output

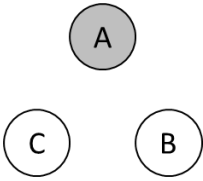
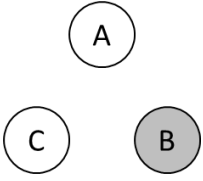
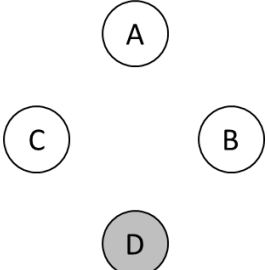
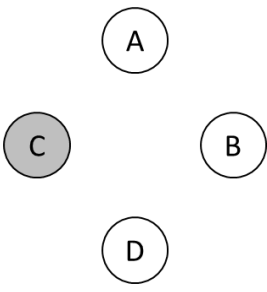
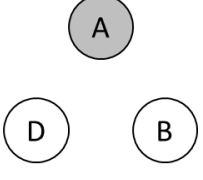
The output should contain **Q** lines, representing the names that are shouted in the order of the events.

### Limits

- $3 \leq N, Q \leq 200,000$
- It is guaranteed that every student’s name will consist of **only English letters** and be at most 10 characters long. However, the names of students might not be distinct. For instance, there can be more than 1 student with the name ‘Bob’.
- It is also guaranteed that there will be at least 3 people in the circle at any point in time.

Sample Input ( <b>ballpassing1.in</b> )	Sample Output ( <b>ballpassing1.out</b> )
3 Alice Bob Charlie 4 NEXT JOIN Donald NEXT LEAVE	Bob Donald Charlie Alice

**Explanation of Sample Testcase**

Event	Visualisation after event
(Starting)	
NEXT	
JOIN Donald	
NEXT	
LEAVE	

**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. You are free to define your own helper methods and classes (or remove existing ones) if it is suitable.
3. Please be reminded that the marking scheme is:
  - a. Public Test Cases (1%)
    - i. 1% for passing **all** test cases, 0% otherwise
  - b. Hidden Test Cases (1%)
    - i. 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)

**Skeleton File – Ballpassing.java**

You are given the skeleton file `Ballpassing.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:

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

import java.util.*;

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

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