题目描述：

幼儿园里有一个放倒的圆桶，它是一个**线性结构**，允许在桶的右边将篮球放入，可以在桶的左边和右边将篮球取出。每个篮球有单独的编号，老师可以**连续放入一个或多个**篮球，小朋友可以在桶左边或右边将篮球取出，当桶里**只有一个篮球的情况下，必须从左边取出**。

如老师按顺序放入 1、2、3、4、5 共 5 个编号的篮球，那么小朋友可以依次取出的编号为“1,2,3,4,5”或者“3,1,2,4,5”编号的篮球，无法取出 “5,1,3,2,4” 编号的篮球

其中“3,1,2,4,5”的取出场景为：连续放入 1,2,3 号 -> 从右边取出 3 号 -> 从左边取出 1 号 -> 从左边取出 2 号 -> 放入 4 号 -> 从左边取出 4 号 -> 放入 5 号 -> 从左边取出 5 号，简单起见，我们以 L 表示左，R 表示右，此时的篮球的依次取出序列为

“ RLLLL ”

输入描述：

每次输入包含一个测试用例：

1、第一行的数字作为老师依次放入的篮球编号；

2、第二行的数字作为要检查是否能够按照放入顺序取出的篮球编号；

其中篮球编号用逗号进行分隔。

输出描述：

对于每个篮球的取出序列，如果确实可以获取，请打印出其按照左右方向的操作的取出顺序，如果无法获取则打印"NO"

补充说明：

1、1<=篮球的编号，篮球个数<=200；

2、篮球上的数字不重复；

3、输出的结果中 LR 的必须为大写；

示例 1

输入：

```
4,5,6,7,0,1,2

6,4,0,1,2,5,7
```

输出：

```
RLRRRLL
```

说明：

篮球的取出顺序依次为“右，左，右，右，右，左，左”

示例 2

输入：

```
4,5,6,7,0,1,2

6,0,5,1,2,4,7
```

输出：

```
NO
```

说明：

无法取出对应序列的篮球

示例 3

输入：

```
1,2,3,4
1,2,3,5
```

输出：

```
NO
```

说明：

不存在编号为 5 的篮球，所以无法取出对应的编号数据

```
import java.util.Scanner;
import java.util.Deque;
import java.util.LinkedList;

// 注意类名必须为 Main, 不要有任何 package xxx 信息
public class Main {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        String input = in.nextLine();
        String output = in.nextLine();
        System.out.println(f(input, output));
    }

    static String f(String input, String output) {
        String[] inputArr = input.split(",");
        String[] outputArr = output.split(",");
        int[] a = new int[inputArr.length];
        int[] b = new int[outputArr.length];

        if (a.length != b.length) {
            return "NO";
        }

        for (int i = 0; i < a.length; i++) {
            a[i] = Integer.parseInt(inputArr[i]);
        }

        for (int i = 0; i < b.length; i++) {
            b[i] = Integer.parseInt(outputArr[i]);
        }

        Deque<Integer> dq = new LinkedList<>();
        StringBuilder res = new StringBuilder();
        int i = 0;
        int j = 0;

        while (i < a.length && j < b.length) {
            dq.addLast(a[i++]);

            while (!dq.isEmpty() && j < b.length) {
                if (b[j] == dq.peekFirst()) {
                    res.append("L");
```

```
                        dq.removeFirst();
                        j++;
                    } else if (b[j] == dq.peekLast()) {
                        res.append("R");
                        dq.removeLast();
                        j++;
                    } else {
                        break;
                    }
                }
            }

            return res.length() == a.length ? res.toString() : "NO";
        }
    }
```