不开心的小朋友题目描述：

游乐场里增加了一批摇摇车，非常受小朋友欢迎，但是每辆摇摇车同时只能有一个小朋友使用，如果没有空余的摇摇车，需要排队等候，或者直接离开，最后没有玩上的小朋友会非常不开心。请根据今天小朋友的来去情况，统计不开心的小朋友数量。

1、摇摇车数量为 N，范围是：1≤N<10；

2、每个小朋友都对应一个编码，编码是不重复的数字，今天小朋友的来去情况，可以使用

编码表示为：1 1 2 3 2 3。（若小朋友离去之前有空闲的摇摇车，则代表玩耍后离开；

不考虑小朋友多次玩的情况）。小朋友数量<=100

3、题目保证所有输入数据无异常且范围满足上述说明。

输入描述：

第一行：摇摇车数量

第二行：小朋友来去情况

输出描述：

返回不开心的小朋友数量

示例 1

输入：

```
1

1 2 1 2
```

输出：

```
0
```

说明：

第一行，1 个摇摇车

第二行，1 号来 2 号来（排队） 1 号走 2 号走（1 号走后摇摇车已有空闲，所以玩后离开）

示例 2

输入：

```
1
1 2 2 3 1 3
```

输出：

```
1
```

说明：

第一行，1 个摇摇车

第二行，1 号来 2 号来（排队） 2 号走（不开心离开） 3 号来（排队） 1 号走 3 号走（1 号走后摇摇车已有空闲，所以玩后离开）

```
import java.util.*;
import java.util.Scanner;

// 注意类名必须为 Main, 不要有任何 package xxx 信息
public class Main {
    public List<Integer> relaxCarList = null;
    public List<Integer> runCarList = null;
    public Map<String, Integer> runCarMap = new HashMap<String, Integer>();
    public Integer maxCarNum = 0;
    public Integer relaxCarNum = 0;
    public Integer runCarNum = 0;
    public List<String>    disAgreeList = new
    ArrayList<String>();//可能会不开心的小朋友
    public Integer disAgreeMapNum = 0;
    public static void main(String[] args) {
        new Main().test03();
    }

    public void test03() {
        Scanner in = new Scanner(System.in);
        // 注意 hasNext 和 hasNextLine 的区别
        while (in.hasNextLine()) { // 注意 while 处理多个 case
            int maxCar = Integer.parseInt(in.nextLine());
            createCarPool(maxCar);//初始化最大车的值
            String ch = in.nextLine();
```

```
            disAgreeMapNum = 0;
            ch = ch.replaceAll(" ", ",");
            String[] children = ch.split(",");
            someMethod(children);
        }
    }

    private void someMethod(String[] children) {
        for (int i = 0; i < children.length; i++) {
            useCar(children[i]);
        }
        System.out.println(disAgreeMapNum);
    }

    private void useCar(String child) {
        if (runCarMap.containsKey(child)) {
            relaxCarList.add(runCarMap.get(child));
            runCarMap.remove(child);
            disAgreeList.remove(child);
            return;
        }

        if (relaxCarList != null && relaxCarList.size() > 0 ) {
            if (!runCarMap.containsKey(
                        child)) { //用车的 map 中不包含该小孩的时候，则去使用该车，
并且空闲车-1
                runCarMap.put(child, relaxCarList.get(0));
                relaxCarList.remove(0);
            } else {
                relaxCarList.add(runCarMap.get(child));
                runCarMap.remove(child);
                disAgreeList.remove(child);
            }
        } else { //用车的 map 中包含该小孩的时候，则该小孩离去,并且空闲车+1
            if (!disAgreeList.contains(child)) {
                disAgreeList.add(child);
            } else {
                disAgreeMapNum++;
                disAgreeList.remove(child);
            }
        }
    }

    /**
```

```
     * 初始化车的信息
     * @param maxCar
     */
    private void createCarPool(int maxCar) {
        maxCarNum = maxCar;
        relaxCarNum = maxCar;
        runCarNum = 0;
        runCarList = new ArrayList<Integer>(maxCar);
        relaxCarList = new ArrayList<Integer>(maxCar);
        for (int i = 0; i < maxCarNum; i++) {
            //给每辆车编号
            relaxCarList.add(i);
        }
    }
}
```