题目描述：
给定一个数组 X 和正整数 K，请找出使表达式 X[i] - X[i + 1] - … - X[i + K - 1]结果最接近于数组中位数的下标 i，如果有多个 i 满足条件，请返回最大的 i。
其中，数组中位数：长度为 N 的数组，按照元素的值大小升序排列后，下标为 N/2 元素的值
补充说明：
1. 数组 X 的元素均为正整数；
2. X 的长度 n 取值范围：2<= n <= 1000；
3. K 大于 0 且小于数组的大小；
4. i 的取值范围：0 <= i < 1000；
5.题目的排序数组 X[N]的中位数是 X[N/2]。

示例 1
输入：
[50,50,2,3],2
输出：
1
说明：
1、中位数为 50：[50,50,2,3]升序排序后变成[2,3,50,50]，中位数为下标 4/2=2 的元素 50；
2、计算结果为 1：X[50,50,2,3]根据题目计算 X[i] - … - X[i + K - 1]得出三个数 0（X[0]-X[1] = 50 - 50）、48（X[1]-X[2 = 50 -2）和 -1（X[2]-X[3] = 2 - 3），其中 48 最接近 50，因此返回下标 1。

```
/**
 * 语句转换
 * @param scores int 整型一维数组 分数
 * @param K int 整型
 * @return int 整型
 */
function findTheStartPosition( scores ,  K ) {
    // write code here
    const arr = []
  for (let i = 0; i < scores.length; i ++) {
    if ((i + K - 1) >= scores.length) break
    let count = scores[i]
    // console.log('i', i)
    for (let j = i + 1; j < i + K; j ++) {
      // console.log('j', j)
      count = count - scores[j]
    }
    arr.push(count)
  }
//   console.log(arr)
  // 算中位数
  let source2 = scores.sort((a, b) => a - b)
```

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
补充说明：

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）

```
const rl = require("readline").createInterface({ input: process.stdin });
var iter = rl[Symbol.asyncIterator]();
const readline = async () => (await iter.next()).value;

void async function () {
```

```
    const tokens = []
    // Write your code here
    while(line = await readline()){
        tokens.push(line)
    }
    const N = +tokens[0]
    const Arr = tokens[1].split(' ').map(i => +i)
//     console.log(N, Arr)

    let waiting = []
  let playing = []
  let cars = N
  let sad = 0
  Arr.forEach(item => {
    let childIndex = waiting.indexOf(item)
    if (childIndex !== -1) {
      // 小朋友伤心离开
      sad ++
      waiting.splice(childIndex, 1)
      return
    }
    childIndex = playing.indexOf(item)
    if (childIndex !== -1) {
      // 小朋友玩完了离开
      cars ++
      playing.splice(childIndex, 1)
      // 需要判断是否有小朋友等待
      if (waiting.length > 0) {
        let child = waiting.shift()
        playing.push(child)
        cars --
        return
      }
      return
    }
    // 小朋友是新来的
    if (cars > 0) {
      cars --
      playing.push(item)
    } else {
      waiting.push(item)
    }
  })
  console.log(sad)
```

}()