实验 2.2　游戏

计科 1903 201914020128 陈旭

一.　问题分析

1). 问题与功能描述

1.需要处理的数据是两个整型(int 型)数字 n(小朋友人数)，k(判断用数)。

2.实现的功能是：先插入队列循环判断作为报数，当所报的数为 k 的倍数或者最后一位是 k 的时候标记为被淘汰并出队列，然后最终只剩一个小朋友未被淘汰的时候输出该小朋友的序号。

3.直接采用标准输出一个整数即可。

2). 样例分析

1. 求解方法：① 将 n 个小朋友，

② 对比每一个元素和它相邻的元素的差值绝对值，用一个数记录最小值，如果新的绝对值比前面的值要小，则记录的数被赋值为新的最小绝对值。最终输出这个用来记录最小值的数。

2. 样例求解：① 【样例一】

输入：5　2
1 号小朋友报数 1;
2 号小朋友报数 2 淘汰;
3 号小朋友报数 3;
4 号小朋友报数 4 淘汰;
5 号小朋友报数 5;
1 号小朋友报数 6 淘汰;
3 号小朋友报数 7;
5 号小朋友报数 8 淘汰;
3 号小朋友获胜。

② 【样例二】

输入：7　3。

1 号小朋友报数 1；
2 号小朋友报数 2；
3 号小朋友报数 3 淘汰；
4 号小朋友报数 4；
5 号小朋友报数 5；
6 号小朋友报数 6 淘汰；
7 号小朋友报数 7；
1 号小朋友报数 8；
2 号小朋友报数 9 淘汰；
4 号小朋友报数 10；
5 号小朋友报数 11；
7 号小朋友报数 12 淘汰；
1 号小朋友报数 13 淘汰；
4 号小朋友报数 14；
5 号小朋友报数 15 淘汰；
4 号小朋友获胜

二.　抽象数据类型设计

需要处理的数据对象的类型为整型，且具有序列性，并且需要多次循环来判断是否被淘汰，我们可以将这一特点归纳为先进先出的特性，因此我们采用 STL 中的 queue(队列)来处理该题目。

三.　算法分析

- **算法思想**：引入 queue 头文件，使用序列式容器 queue 进行数据的存储。然后依次压入队列,循环报数，如果符合上述两个条件之一的采用出队列操作；不满足则重新压入队尾进行上述操作。

- **样例分析**：① 【样例一】

  输入：5　2

  1 号小朋友报数 1，弹出队列后压入队尾；
  2 号小朋友报数 2 出队列；
  3 号小朋友报数 3，弹出队列后压入队尾；
  4 号小朋友报数 4 出队列；
  5 号小朋友报数 5，弹出队列后压入队尾；
  1 号小朋友报数 6 出队列；
  3 号小朋友报数 7，弹出队列后压入队尾；
  5 号小朋友报数 8 出队列；
  3 号小朋友获胜，输出 3。

  ② 【样例二】

输入：7　3
1 号小朋友报数 1，弹出队列后压入队尾；
2 号小朋友报数 2，弹出队列后压入队尾；
3 号小朋友报数 3，出队列；
4 号小朋友报数 4，弹出队列后压入队尾；
5 号小朋友报数 5，弹出队列后压入队尾；
6 号小朋友报数 6，出队列；
7 号小朋友报数 7，弹出队列后压入队尾；
1 号小朋友报数 8，弹出队列后压入队尾；
2 号小朋友报数 9，出队列；
4 号小朋友报数 10，弹出队列后压入队尾；
5 号小朋友报数 11，弹出队列后压入队尾；
7 号小朋友报数 12，出队列；
1 号小朋友报数 13，出队列；
4 号小朋友报数 14，弹出队列后压入队尾；
5 号小朋友报数 15，出队列；
4 号小朋友获胜，输出 4。

- **关键功能的算法步骤：**关键步骤实现在一个 while 循环里。

```
while(listsize<1)
{
    num=list.front(); //记录队首元素
    if(order.lastnumber!=k&&order%k!=0) //判断是否满足游戏规则
    {
        list.push(num); //不能被淘汰的继续插入队尾。
    }
    order++;//报数加一
}
```

- **性能分析**

【空间复杂度】所调用的 queue 的内部函数，其空间复杂度为 C_1,循环中每往队列里面插入一个数，进行一次空间占用，开销为 C_2n;故整个程序的总空间复杂度为 O(n)

【时间复杂度】先循环输入值并引入队列，时间开销为 C_1n，然后循环判断，分析知 k 次循环以内必然判断结束，则时间开销为 C_2n，综上时间复杂度为 O(n)。