**算法训练 筛选号码**

时间限制：1.0s   内存限制：512.0MB

问题描述

　　有n个人围成一圈，顺序排号（编号为1到n）。从第1个人开始报数(从1到3报数)，凡报到3的人退出圈子。从下一个人开始继续报数，直到剩下最后一个人，游戏结束。  
　　问最后留下的是原来第几号的那位。  
　　举个例子，8个人围成一圈：  
　　1 2 3 4 5 6 7 8  
　　第1次报数之后，3退出，剩下：  
　　1 2 4 5 6 7 8 （现在从4开始报数）  
　　第2次报数之后，6退出，剩下：  
　　1 2 4 5 7 8 （现在从7开始报数）  
　　第3次报数之后，1退出，剩下：  
　　2 4 5 7 8 （现在从2开始报数）  
　　第4次报数之后，5退出，剩下：  
　　2 4 7 8 （现在从7开始报数）  
　　第5次报数之后，2退出，剩下：  
　　4 7 8 （现在从4开始报数）  
　　第6次报数之后，8退出，剩下：  
　　4 7 （现在从4开始报数）  
　　最后一次报数之后，4退出，剩下：  
　　7.  
　　所以，最后留下来的人编号是7。

输入格式

　　一个正整数n，(1<n<10000)

输出格式

　　一个正整数，最后留下来的那个人的编号。

样例输入

8

样例输出

7

数据规模和约定

　　对于100%的数据，1<n<10000。

本题的C++参考代码如下：

#include<stdio.h>

main()

{ int i,N,x,m,s;

x=3;

s=0;

scanf("%d",&m);

for(i=2;i<=m;i++)

s=(s+x)%i;

printf("%d\n",s+1);

}

本题的C参考代码如下：

#include <stdio.h>

#include <malloc.h>

typedef struct form{

int num;

int date;

struct form \*link;}

del;

del \*creat(int n){

int i;

del \*head,\*p1,\*p2;

head=(del \*)malloc(sizeof(del));

p1=(del \*)malloc(sizeof(del));

head->link=p1;

for(i=1;i<=n-2;i++){p2=(del \*)malloc(sizeof(del));

p1->link=p2;

p1=p2;}

p1->link=head;

return(head);

}

void dateop(del \*h,int n){

del \*p;

int i,j=1;

p=h;

for(i=1;i<=n;i++){

p->num=i;

p->date=j;j++;

if(j==4) j=1;

p=p->link;}

}

int deal(del \*h,int n){

del \*k;

int s;

int count,j=1,i;

count=n;

k=h;

while(count!=1){

if(j==3&&k->date!=0) {k->date=0;count--;}

k=k->link;

j++;

if(k->date==0) j--;

if(j==4) j=1;

}

k=h;

for(i=1;i<=n;i++){

if(k->date!=0) {s=k->num;break;}

k=k->link;}

return(s);}

int main(){

int x;

int i;

del \*p;

scanf("%d",&x);

p=creat(x);

dateop(p,x);

x=deal(p,x);

for(i=1;i<x;i++){

p=p->link;}

printf("%d",p->num);

return 0;

}

本题的Java参考代码如下

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.util.ArrayList;

public class Main {

public static void main(String[] args) throws Exception {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

int length = Integer.parseInt(br.readLine());

ArrayList<Integer> arr = new ArrayList<Integer>();

for (int i = 1; i <= length; i++) {

arr.add(i);

}

int control = 0;

int index = 0;

while (arr.size() != 1) {

if (control == 2) {

arr.remove(index);

index--;

}

control = (control + 1) % 3;

index = (index + 1) % arr.size();

}

System.out.println(arr.get(0));

}

}