**问题描述**

|  |  |
| --- | --- |
| 试题编号： | 201803-2 |
| 试题名称： | 碰撞的小球 |
| 时间限制： | 1.0s |
| 内存限制： | 256.0MB |
| 问题描述： | **问题描述**  　　数轴上有一条长度为L（L为偶数)的线段，左端点在原点，右端点在坐标L处。有n个不计体积的小球在线段上，开始时所有的小球都处在偶数坐标上，速度方向向右，速度大小为1单位长度每秒。 　　当小球到达线段的端点（左端点或右端点）的时候，会立即向相反的方向移动，速度大小仍然为原来大小。 　　当两个小球撞到一起的时候，两个小球会分别向与自己原来移动的方向相反的方向，以原来的速度大小继续移动。 　　现在，告诉你线段的长度L，小球数量n，以及n个小球的初始位置，请你计算t秒之后，各个小球的位置。  **提示**  　　因为所有小球的初始位置都为偶数，而且线段的长度为偶数，可以证明，不会有三个小球同时相撞，小球到达线段端点以及小球之间的碰撞时刻均为整数。 　　同时也可以证明两个小球发生碰撞的位置一定是整数（但不一定是偶数）。  **输入格式**  　　输入的第一行包含三个整数n, L, t，用空格分隔，分别表示小球的个数、线段长度和你需要计算t秒之后小球的位置。 　　第二行包含n个整数a1, a2, …, an，用空格分隔，表示初始时刻n个小球的位置。  **输出格式**  　　输出一行包含n个整数，用空格分隔，第i个整数代表初始时刻位于ai的小球，在t秒之后的位置。  **样例输入**  3 10 5 4 6 8  **样例输出**  7 9 9  **样例说明**  　　初始时，三个小球的位置分别为4, 6, 8。 http://118.190.20.162/RequireFile.do?fid=b6beJN6e 　　一秒后，三个小球的位置分别为5, 7, 9。 http://118.190.20.162/RequireFile.do?fid=Ab8QmfeR 　　两秒后，第三个小球碰到墙壁，速度反向，三个小球位置分别为6, 8, 10。 http://118.190.20.162/RequireFile.do?fid=fgQLYbNn 　　三秒后，第二个小球与第三个小球在位置9发生碰撞，速度反向（注意碰撞位置不一定为偶数），三个小球位置分别为7, 9, 9。 http://118.190.20.162/RequireFile.do?fid=erfyNJDT 　　四秒后，第一个小球与第二个小球在位置8发生碰撞，速度反向，第三个小球碰到墙壁，速度反向，三个小球位置分别为8, 8, 10。 http://118.190.20.162/RequireFile.do?fid=m5EBf6q8 　　五秒后，三个小球的位置分别为7, 9, 9。 http://118.190.20.162/RequireFile.do?fid=JRHaHt4T  **样例输入**  10 22 30 14 12 16 6 10 2 8 20 18 4  **样例输出**  6 6 8 2 4 0 4 12 10 2  **数据规模和约定**  　　对于所有评测用例，1 ≤ n ≤ 100，1 ≤ t ≤ 100，2 ≤ L ≤ 1000，0 < ai < L。L为偶数。 　　保证所有小球的初始位置互不相同且均为偶数。 |

思路：1. 貌似暴力模拟，来一发试试。也好像有简单思路，但还没有成型的思路，先来模拟吧。

代码：

import java.text.DecimalFormat;

import java.io.\*;

import java.util.\*;

import java.math.\*;

public class Main{

static Scanner in=new Scanner(System.in);

static int n=0,t=0,L=0;

static int[] pos=new int[1000];

static boolean[] der=new boolean[1000];

static int[] to=new int[1000];

public static void main(String[] args) {

n=in.nextInt();

L=in.nextInt();

t=in.nextInt();

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

pos[i]=in.nextInt();

to[i]=i;

}

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

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

if(pos[i]>pos[j]) {

int temp=pos[i];

pos[i]=pos[j];

pos[j]=temp;

temp=to[i];

to[i]=to[j];

to[j]=temp;

}

}

}

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

for(int j=1;j<=n;++j) {

if(der[j]) {

pos[j]--;

}else {

pos[j]++;

}

}

if(pos[1]==0) {

der[1]=!der[1];

}

if(pos[n]==L) {

der[n]=!der[n];

}

for(int j=2;j<=n;++j) {

if(pos[j]==pos[j-1]) {

der[j]=!der[j];

der[j-1]=!der[j-1];

}

}

}

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

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

if(to[i]>to[j]) {

int temp=pos[i];

pos[i]=pos[j];

pos[j]=temp;

temp=to[i];

to[i]=to[j];

to[j]=temp;

}

}

}

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

System.out.print(pos[i]+" ");

}

System.out.println(pos[n]);

}

}

