图片包含 屏幕截图

描述已自动生成





import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner input=new Scanner(System.in);

int n=input.nextInt(),m=input.nextInt();

char[] A=new char[n+m-1];//定义数组A

int Aindex=n-1;//确定字符'A'的位置为n-1

for(int i=Aindex;i>=0;--i)//填充数组A的0~Aindex-1位置的字符

A[i]=(char)('A'+Aindex-i);

for(int i=Aindex;i<n+m-1;++i)//填充数组A的Aindex+1~m+n-2位置的字符

A[i]=(char)('A'+i-Aindex);

for(int i=0;i<n;++i){//输出n行字符串

for(int j=0;j<m;++j)

System.out.print(A[Aindex-i+j]);

System.out.println();//换行

}

}

}

​

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner input=new Scanner(System.in);

long N=input.nextLong();

System.out.println((1+N)\*N/2);

}

}

图片包含 屏幕截图

描述已自动生成

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

double r = input.nextDouble();

System.out.printf("%.7f", Math.PI \* r \* r);//java的Math类中已经定义了圆周率PI

}

}

图片包含 屏幕截图

描述已自动生成

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

int N = input.nextInt(), Fn = 1, Fn1 = 0;//Fn来表示当前的斐波那契数值，Fn1用来表示Fn的上一个斐波那契数值

for (int i = 1; i < N; ++i) {//进行n-1次循环

int temp = (Fn + Fn1) % 10007;//temp为以Fn和Fn1为前两项的斐波那契数对10007的余数

Fn1 = Fn;

Fn = temp;//Fn和Fn1均指向下一个斐波那契数

}

System.out.println(Fn);

}

}

图片包含 屏幕截图

描述已自动生成

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

int y=input.nextInt();

System.out.println((y%4==0&&y%100!=0)||y%400==0?"yes":"no");

}

}

图片包含 屏幕截图

描述已自动生成

public class Main {

public static void main(String[] args) {

for(int a=0;a<2;++a)

for(int b=0;b<2;++b)

for(int c=0;c<2;++c)

for(int d=0;d<2;++d)

for(int e=0;e<2;++e)

System.out.printf("%d%d%d%d%d\n",a,b,c,d,e);

}

}



import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner input=new Scanner(System.in);

int n=input.nextInt(),k1=Integer.MIN\_VALUE,k2=Integer.MAX\_VALUE,k3=0;

for(int i=0;i<n;++i){

int t=input.nextInt();

k1=Math.max(k1, t);

k2=Math.min(k2, t);

k3+=t;

}

System.out.println(k1+"\n"+k2+"\n"+k3);

}

}

图片包含 屏幕截图

描述已自动生成

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

int n = input.nextInt();

int[] A = new int[n];

for (int i = 0; i < n; ++i)

A[i] = input.nextInt();

int m = input.nextInt(), ans = -1;

for (int i = 0; i < n; ++i)

if (A[i] == m){

ans = i + 1;

break;

}

System.out.println(ans);

}

}

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

图片包含 屏幕截图

描述已自动生成

int n = in.nextInt();

int[][] a = new int[n][n];// 存储杨辉三角

for (int i = 0; i < n; ++i) {

for (int j = 0; j <= i; ++j) {

if (j == 0 || j == i)

a[i][j] = 1;

else

a[i][j] = a[i - 1][j - 1] + a[i - 1][j];

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

}

System.out.println();

}

}

}

图片包含 屏幕截图

描述已自动生成

public class Main {

public static void main(String[] args) {

for (int i = 101; i < 1000; ++i) {

int k1 = i % 10, k2 = i % 100 / 10, k3 = i / 100;

if (i == k1 \* k1 \* k1 + k2 \* k2 \* k2 + k3 \* k3 \* k3)

System.out.println(i);

}

}

}

图片包含 屏幕截图

描述已自动生成

public class Main {

public static void main(String[] args) {

for(int i=1001;i<10000;++i)

if(i/1000==i%10&&(i/10%10)==(i/100%10))

System.out.println(i);

}

}

图片包含 屏幕截图

描述已自动生成

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner input=new Scanner(System.in);

int n=input.nextInt();

for(int i=1;i<10;++i)

for(int j=0;j<10;++j)

for(int k=0;k<10;++k)

if(2\*i+2\*j+k==n)//验证五位数

System.out.println(i\*10000+j\*1000+k\*100+j\*10+i);

for(int i=1;i<10;++i)

for(int j=0;j<10;++j)

for(int k=0;k<10;++k)

if(2\*i+2\*j+2\*k==n)//验证六位数

System.out.println(i\*100000+j\*10000+k\*1000+k\*100+j\*10+i);

}

}

图片包含 屏幕截图

描述已自动生成

import java.util.\*;

public class Main {

public static void main(String[] args) {

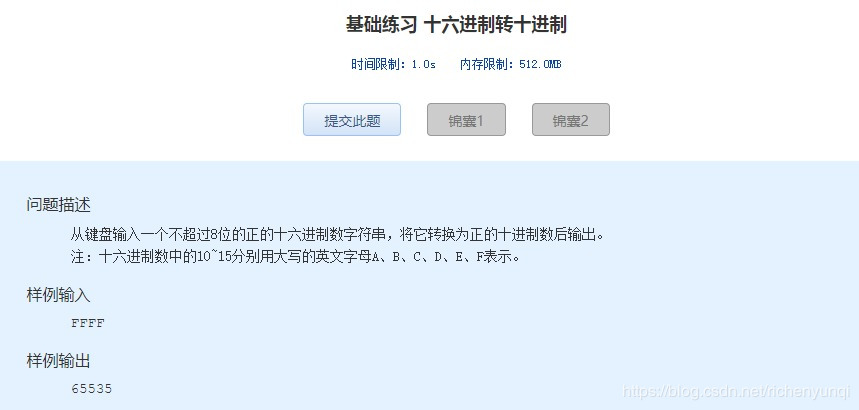
Scanner input=new Scanner(System.in);

int n=input.nextInt();

System.out.printf("%X", n);

}

}



import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner input=new Scanner(System.in);

String s=input.nextLine();

System.out.println(Long.valueOf(s,16));

}

}



import java.io.\*;

public class Main {

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

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

int n=Integer.parseInt(input.readLine());

for(int ii=0;ii<n;++ii){

char[] s1=input.readLine().toCharArray();//s1读取16进制字符串

StringBuilder s2=new StringBuilder();//s2存储转换后的2进制字符串

for(int i=0;i<s1.length;++i)//转换成2进制字符串

switch(s1[i]){

case '0':s2.append("0000");break;

case '1':s2.append("0001");break;

case '2':s2.append("0010");break;

case '3':s2.append("0011");break;

case '4':s2.append("0100");break;

case '5':s2.append("0101");break;

case '6':s2.append("0110");break;

case '7':s2.append("0111");break;

case '8':s2.append("1000");break;

case '9':s2.append("1001");break;

case 'A':s2.append("1010");break;

case 'B':s2.append("1011");break;

case 'C':s2.append("1100");break;

case 'D':s2.append("1101");break;

case 'E':s2.append("1110");break;

case 'F':s2.append("1111");break;

}

while(s2.length()%3!=0)//如果s2的长度不是3的倍数，在s2字符串前补足够的0

s2.insert(0, "0");

s1=s2.toString().toCharArray();

int i=0;

while(i<s1.length&&(s1[i]-'0')\*4+(s1[i+1]-'0')\*2+(s1[i+2]-'0')==0) i+=3;//前导0不输出

if(i==s1.length) System.out.println("0");//如果i已经走到2进制字符串末尾，说明字符串表示的数字是0，输出0

else

for(;i<s1.length;i+=3)//3位一输出

System.out.print((s1[i]-'0')\*4+(s1[i+1]-'0')\*2+(s1[i+2]-'0'));

System.out.println();

}

}

}

图片包含 屏幕截图

描述已自动生成

import java.util.\*;

public class Main {

public static void main(String[] args){

Scanner input = new Scanner(System.in);

int N = input.nextInt();

int[] a = new int[N];

for (int i = 0; i < N; ++i)

a[i] = input.nextInt();

Arrays.sort(a);

for (int i = 0; i < N; ++i)

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

}

}



import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner input=new Scanner(System.in);

int N=input.nextInt();

int []A=new int[N+1];

for(int i=1;i<=N;++i)

A[i]=input.nextInt();

int M=input.nextInt();

for(int i=0;i<M;++i){

int L=input.nextInt(),R=input.nextInt(),K=input.nextInt();

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

for(int j=L;j<=R;++j)

list.add(A[j]);

Collections.sort(list,Collections.reverseOrder());//从大到小进行排序

System.out.println(list.get(K-1));

}

}

}

图片包含 屏幕截图

描述已自动生成

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

long N = input.nextLong();

if (N % 2 == 1) {

System.out.println(N \* (N - 1) \* (N - 2));

} else if (N % 3 != 0)

System.out.println(N \* (N - 1) \* (N - 3));

else

System.out.println((N - 1) \* (N - 2) \* (N - 3));

}

}

图片包含 屏幕截图

描述已自动生成

import java.io.\*;

import java.util.\*;

class Edge {// 边的结构体

int v, cost;

Edge(int vv, int c) {v = vv;cost = c;}

};

public class Main {

public static int[] dis = new int[20005];// dis表示从源点到其他顶点的距离

public static ArrayList<ArrayList<Edge>> graph = new ArrayList<ArrayList<Edge>>();

public static boolean[] inQueue = new boolean[20005];// 顶点是否在队列中

public static int[] nextInt(BufferedReader input) throws IOException {// 读取下一行的整数

String[] s = input.readLine().split(" ");

int[] integer = new int[s.length];

for (int i = 0; i < s.length; ++i)

integer[i] = Integer.parseInt(s[i]);

return integer;

}

public static void shortestDis() {// 最短路径算法

Queue<Integer> q = new LinkedList<>();

q.add(1);// 源点入队

inQueue[1] = true;// 源点在队列中，置inQueue为true

while (!q.isEmpty()) {

int t = q.poll();// 一个顶点出队

inQueue[t] = false;// 顶点不在队中，置inQueue为false

for (Edge e : graph.get(t))

if (dis[t] + e.cost < dis[e.v]) {// 如果最短距离可更新

dis[e.v] = dis[t] + e.cost;// 更新最短距离

if (!inQueue[e.v]) {// 不在队列中

q.add(e.v);// 入队

inQueue[e.v] = true;// 在队中，置inQueue为true

}

}

}

}

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

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

int[] NM = nextInt(input);

for (int i = 0; i <= NM[0]; ++i)

graph.add(new ArrayList<>());

for (int i = 0; i < NM[1]; ++i) {

int[] a = nextInt(input);

graph.get(a[0]).add(new Edge(a[1], a[2]));

}

Arrays.fill(dis, Integer.MAX\_VALUE);

dis[1] = 0;

shortestDis();

for (int i = 2; i < NM[0] + 1; ++i)

System.out.println(dis[i]);

}

}



import java.io.\*;

import java.util.\*;

class Edge {//边类

int v1, v2, cost;

Edge(int vv1, int vv2, int c) {//构造函数

v1 = vv1;v2 = vv2;cost = c;

}

}

public class Main {

public static int[] cost = new int[10005], father = new int[10005];// N个顶点，P条边以及点权，并查集

public static PriorityQueue<Edge> edges = new PriorityQueue<>(new Comparator<Edge>() {//定义比较器

public int compare(Edge o1, Edge o2) {//比较函数

return o1.cost - o2.cost;

}

});// 边的优先级队列

public static int[] nextInt(BufferedReader input) throws IOException {//读取下一行的整数

String[] s = input.readLine().split(" ");

int[] integer = new int[s.length];

for (int i = 0; i < s.length; ++i)

integer[i] = Integer.parseInt(s[i]);

return integer;

}

public static int findFather(int x) {// 查找父亲结点并进行路径压缩

if (x == father[x])

return x;

int temp = findFather(father[x]);

father[x] = temp;

return temp;

}

public static int Kruskal() {// 求解最小生成树的权值之和

int sumCost = 0; // sumCost表示整棵最小生成树的各边权值之和

while (!edges.isEmpty()) {

Edge e = edges.poll();// 弹出当前队列中边权最小的边

int ua = findFather(e.v1), ub = findFather(e.v2);

if (ua != ub) {

sumCost += e.cost;

father[ua] = ub;

}

}

return sumCost;// 返回整棵树的权值之和

}

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

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

int []NP=nextInt(input);

for (int i = 1; i <= NP[0]; ++i) {

father[i] = i;// 初始化并查集

cost[i] = nextInt(input)[0];

}

for (int i = 0; i < NP[1]; ++i) {

int []a=nextInt(input);

edges.add(new Edge(a[0], a[1], 2 \* a[2] + cost[a[0]] + cost[a[1]]));// 更新成新的边权，并加入优先级队列

}

int sumCost = Kruskal(), k = Integer.MAX\_VALUE;// 求得最小生成树的权值之和

for (int i = 1; i < NP[0] + 1; ++i)// 计算最小的点权

k = Math.min(cost[i], k);

System.out.println(sumCost + k);

}

}



import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

int N=input.nextInt(),M=input.nextInt();

int [][]A=new int[N][M];

for(int i=0;i<M;++i){

int a=input.nextInt(),b=input.nextInt();

A[a-1][i]=1;

A[b-1][i]=-1;

}

for(int i=0;i<N;++i){

for(int j=0;j<M;++j)

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

System.out.println();

}

}

}

图片包含 屏幕截图

描述已自动生成

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

int N=input.nextInt(),index=-1,MAX=Integer.MIN\_VALUE;

for(int i=0;i<N;++i){

int in=input.nextInt();

if(in>MAX){

MAX=in;

index=i;

}

}

System.out.println(MAX+" "+index);

}

}

图片包含 屏幕截图

描述已自动生成

import java.util.\*;

public class Main {

public static ArrayList<Integer> prime=new ArrayList<>();

public static boolean []flag=new boolean[(int)(1e6+5)];

public static void findPrime(){

for(int i=2;i<flag.length;++i)

if(!flag[i]){

prime.add(i);

for(int j=i+i;j<flag.length;j+=i)

flag[j]=true;

}

}

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

findPrime();

int N=input.nextInt(),result=1;

for(int i=0;i<N;++i)

result=(result\*prime.get(i))%50000;

System.out.println(result);

}

}



import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

int N=input.nextInt();

for(int i=0;i<N;++i){

int M=input.nextInt(),sum=0;

int []A1=new int[M],A2=new int[M];

for(int j=0;j<M;++j)

A1[j]=input.nextInt();

for(int j=0;j<M;++j)

A2[j]=input.nextInt();

Arrays.sort(A1);

Arrays.sort(A2);

for(int j=0;j<M;++j)

sum+=A1[j]\*A2[M-j-1];

System.out.println(sum);

}

}

}

图片包含 屏幕截图

描述已自动生成

import java.util.\*;

public class Main {

public static int CompactIntegers(int A[],int N){

for(int i=N-1;i>=0;--i)//从后向前遍历数组

if(A[i]==0){//遇到了0,此位置之后所有数字均向前移动一位

for(int j=i;j<N-1;++j)

A[j]=A[j+1];

--N;//数组长度递减

}

return N;//返回当前数组长度

}

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

int N=input.nextInt();

int []A=new int[N];

for(int i=0;i<N;++i)

A[i]=input.nextInt();

N=CompactIntegers(A,N);

System.out.println(N);

for(int i=0;i<N;++i)

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

}

}

图片包含 屏幕截图

描述已自动生成

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

char []s=input.nextLine().toCharArray();

for(char c:s)

if(Character.isUpperCase(c))//是大写字母

System.out.print(Character.toLowerCase(c));//转换成小写

else if(Character.isLowerCase(c))//是小写字母

System.out.print(Character.toUpperCase(c));//转换为大写

}

}

图片包含 屏幕截图

描述已自动生成

import java.io.\*;

public class Main {

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

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

String[] s = input.readLine().split(" ");//将读取的一行字符串按空格分割成几个字符串

int M = Integer.parseInt(s[0]), S = Integer.parseInt(s[1]), N = Integer.parseInt(s[2]);//将字符串转换为整数

int[][] A1 = new int[M][S], A2 = new int[S][N], A3 = new int[M][N];

for (int i = 0; i < M; ++i) {

s = input.readLine().split(" ");//将读取的一行字符串按空格分割成几个字符串

for (int j = 0; j < S; ++j)

A1[i][j] = Integer.parseInt(s[j]);//将字符串转换为整数

}

for (int i = 0; i < S; ++i){

s = input.readLine().split(" ");//将读取的一行字符串按空格分割成几个字符串

for (int j = 0; j < N; ++j)

A2[i][j] = Integer.parseInt(s[j]);//将字符串转换为整数

}

for (int i = 0; i < M; ++i)

for (int j = 0; j < N; ++j) {

A3[i][j] = 0;

for (int k = 0; k < S; ++k)

A3[i][j] += A1[i][k] \* A2[k][j];

}

for (int i = 0; i < M; ++i) {

for (int j = 0; j < N; ++j)

System.out.print(+A3[i][j] + " ");

System.out.println();

}

}

}

图片包含 屏幕截图

描述已自动生成

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

int L = input.nextInt();

String s = input.nextLine();// 读取换行符

s = input.nextLine();

HashMap<String, Integer> m = new HashMap<>();//存储对应子串及其出现次数

ArrayList<String> v = new ArrayList<>();//存储所有子串，注意v中字符串就是以出现时间前后顺序排列的

for (int i = 0; i < s.length(); ++i)// 遍历所有大于等于L的子串

for (int j = i + L; j < s.length() + 1; ++j) {

String sub = s.substring(i, j);

v.add(sub);// 找到某一子串

m.put(sub, m.containsKey(sub) ? m.get(sub) + 1 : 1);//递增出现次数

}

int ans = 0;// 最终结果字符串在v中索引

for (int i = 1; i < v.size(); ++i)// 遍历整个v找到最符合条件的子串

if (m.get(v.get(i)) > m.get(v.get(ans))

|| (m.get(v.get(i)) == m.get(v.get(ans)) && v.get(i).length() > v.get(ans).length()))

ans = i;// 当前指向的字符串相关指标比temp更优,更新ans

System.out.println(v.get(ans));

}

}

图片包含 屏幕截图

描述已自动生成

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

int N=input.nextInt();

if(N>0){

Map<Integer,Integer>map=new TreeMap<Integer, Integer>();

for(int i=0;i<N;++i){

int a=input.nextInt();

map.put(a, map.containsKey(a)?map.get(a)+1:1);

}

int ans=Integer.MAX\_VALUE;

for(Map.Entry<Integer, Integer> entry:map.entrySet())

if(ans==Integer.MAX\_VALUE||entry.getValue()>map.get(ans))

ans=entry.getKey();

System.out.println(ans);

}

}

}

图片包含 屏幕截图

描述已自动生成

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

char[] s1 = input.nextLine().toCharArray(), s2 = input.nextLine().toCharArray();

if (s1.length != s2.length) {// 如果字符串长度不等，输出N

System.out.println("N");

return;

}

int[] hashTable = new int[128];// 定义辅助数组

for (int i = 0; i < s1.length; ++i) {

++hashTable[Character.toLowerCase(s1[i])];// 将hashTable数组的对应位置的元素+1

--hashTable[Character.toLowerCase(s2[i])];// 将hashTable数组的对应位置的元素-1

}

for (int i = 0; i < 128; ++i)// 遍历hashTable数组查找是否有元素不为0

if (hashTable[i] != 0) {// 元素不为0输出N

System.out.println("N");

return;

}

System.out.println("Y");// 输出Y

}

}

图片包含 屏幕截图

描述已自动生成

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

char c1=input.next().charAt(0);

int c2=input.nextInt(),c3=input.nextInt();

if(c1=='+')

System.out.println(c2+c3);

else if(c1=='-')

System.out.println(c2-c3);

else if(c1=='\*')

System.out.println(c2\*c3);

else

System.out.println(c2/c3);

}

}

图片包含 屏幕截图

描述已自动生成

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

int []A=new int[3];

for(int i=0;i<3;++i)

A[i]=input.nextInt();

Arrays.sort(A);

for(int i=2;i>=0;--i)

System.out.printf("%d ",A[i]);

}

}

图片包含 屏幕截图

描述已自动生成

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

int n = input.nextInt();

for (int i = n; i > 0; --i) {

for (int j = 0; j < i; ++j)

System.out.print("\* ");

System.out.println();

}

}

}

图片包含 屏幕截图

描述已自动生成

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

char k=input.next().charAt(0);

double a=input.nextDouble(),b=input.nextDouble(),c=input.nextDouble(),d=input.nextDouble(),m,n;

if(k=='+'){

m = a + c;

n = b + d;

}else if(k=='-'){

m = a - c;

n = b - d;

}else if(k=='\*'){

m = a \* c - b \* d;

n = a \* d + b \* c;

}else{

double kk = c \* c + d \* d;

m = (a \* c + b \* d) / kk;

n = (b \* c - a \* d) / kk;

}

System.out.printf("%.2f+%.2fi",m,n);

}

}

图片包含 屏幕截图

描述已自动生成

import java.util.\*;

public class Main {

public static int gcd(int a,int b){//欧几里得算法求最大公约数

return b==0?a:gcd(b,a%b);

}

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

int a=input.nextInt(),b=input.nextInt();

System.out.println(a/gcd(a,b)\*b);

}

}

图片包含 屏幕截图

描述已自动生成

import java.util.\*;

public class Main {

public static int f(int k,int n){

if(k==0||k==n)

return 1;

return f(k-1,n-1)+f(k,n-1);

}

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

int a=in.nextInt(),b=in.nextInt();

System.out.println(f(a,b));

}

}

图片包含 屏幕截图

描述已自动生成

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

char c=in.next().charAt(0);

System.out.println((char)(c+32));

}

}

图片包含 屏幕截图

描述已自动生成

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

double a=input.nextDouble(),b=input.nextDouble();

double ai=a/(Math.sqrt(a\*a+b\*b));

double bi=b/(Math.sqrt(a\*a+b\*b));

System.out.printf("%.1f+%.1fi\n",ai,bi);

}

}

图片包含 屏幕截图

描述已自动生成

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

int i=input.nextInt();

System.out.printf("%o",i);

}

}



import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

int n=input.nextInt(),sum=0;

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

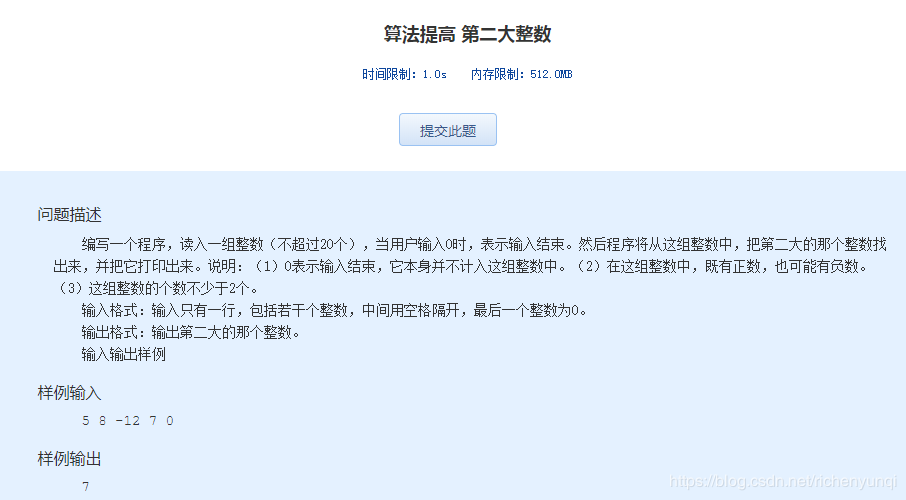
if(n%i==0)

++sum;

System.out.println(sum);

}

}



import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

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

for(int a=input.nextInt();a!=0;a=input.nextInt())

num.add(a);

Collections.sort(num);;//进行从小到大排序

System.out.println(num.get(num.size()-2));

}

}

图片包含 屏幕截图

描述已自动生成

import java.util.\*;

public class Main{

public static void main(String[] args){

Scanner input=new Scanner(System.in);

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

for(int a=input.nextInt();a!=0;a=input.nextInt())//读取数据

list.add(a);

Collections.reverse(list);//将序列翻转

for(int i:list)

System.out.print(i+" ");

}

}

图片包含 屏幕截图

描述已自动生成

import java.text.DecimalFormat;

import java.util.\*;

public class Main{

public static void main(String[] args){

Scanner input=new Scanner(System.in);

double A1=input.nextDouble(),B1=input.nextDouble();

char c=input.next().charAt(0);

double A2=input.nextDouble(),B2=input.nextDouble(),A3=0,B3=0;

if(c=='+'){

A3=A1+A2;

B3=B1+B2;

}else if(c=='-'){

A3=A1-A2;

B3=B1-B2;

}else if(c=='\*'){

A3=A1\*A2-B1\*B2;

B3=A1\*B2+B1\*A2;

}else if(c=='/'){

if(A2==0&&B2==0){

System.out.println("error");

return;

}

A3=(A1\*A2+B1\*B2)/(A2\*A2+B2\*B2);

B3=(B1\*A2-A1\*B2)/(A2\*A2+B2\*B2);

}

DecimalFormat format = new DecimalFormat("#0.#######");//DecimalForma类进行格式化输出

System.out.print(format.format(A3)+(B3>0?"+":""));

if(B3!=0)

System.out.print(format.format(B3)+"i");

}

}

图片包含 屏幕截图

描述已自动生成

import java.util.\*;

public class Main{

public static void main(String[] args){

Scanner input=new Scanner(System.in);

char[] s=input.nextLine().toCharArray();

int ans=1;//字符串周期

while(true){

while(s.length%ans!=0)//若ans不能整除字符串长度，则递增ans

++ans;

boolean f=true;

for(int i=ans;i<s.length&&f;i+=ans)//进行N/ans重循环

for(int j=0;j<ans&&f;++j)//判断i~ans-1字符和i~i+ans-1个字符是否完全一致

if(s[j]!=s[j+i]){//字符不同

++ans;//递增ans

f=false;

}

if(f)//N/ans重循环字符均一致，跳出循环

break;

}

System.out.println(ans);

}

}

图片包含 屏幕截图

描述已自动生成

import java.util.\*;

public class Main{

public static void main(String[] args){

Scanner input=new Scanner(System.in);

int n=input.nextInt(),q=input.nextInt();

int []A=new int[n];

for(int i=0;i<n;++i)

A[i]=input.nextInt();

for(int i=0;i<q;++i){

int left=input.nextInt(),right=input.nextInt(),k=A[left];

for(int j=left+1;j<=right;++j)

k=Math.min(A[j],k);

System.out.println(k);

}

}

}

图片包含 屏幕截图, 文字

描述已自动生成

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

int n = in.nextInt();

PriorityQueue<Integer> p=new PriorityQueue<>();//优先级队列，保存未创建的文档编号

for(int i=0;i<n;++i)//将1~n这n个整数压入优先级队列中

p.add(i+1);

for (int i = 0; i < n; ++i) {

String s=in.next();

if(s.equals("New"))//新建文档

System.out.println(p.poll());//输出优先级队列队首元素编号，并弹出该元素

if(s.equals("Delete")){//删除文档

int a=in.nextInt();

if(p.contains(a))//该文档在优先级队列中，即并未创建，不可删除

System.out.println("Failed");//输出Failed

else{//否则

System.out.println("Successful");//输出Successful

p.add(a);//该文档被删除，将该文档编号压入优先级队列

}

}

}

}

}



import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

int n = input.nextInt(),MAX=Integer.MIN\_VALUE,MIN=Integer.MAX\_VALUE;

for(int i=0;i<n;++i){

int a=input.nextInt();

MAX=Math.max(MAX, a);

MIN=Math.min(MIN, a);

}

System.out.println(MAX+" "+MIN);

}

}

图片包含 屏幕截图

描述已自动生成

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

String s=input.nextLine();

int []weight={7,9,10,5,8,4,2,1,6,3,7,9,10,5,8,4,2};

String trans="10x98765432";

s=s.substring(0,6)+"19"+s.substring(6);

int sum=0;

for(int i=0;i<s.length();++i)

sum+=(s.charAt(i)-'0')\*weight[i];

s+=trans.charAt(sum%11);

System.out.println(s);

}

}

图片包含 屏幕截图

描述已自动生成

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

int []a=new int[3];

a[0]=input.nextInt();a[1]=input.nextInt();a[2]=input.nextInt();

Arrays.sort(a);

System.out.println(a[2]+" "+a[1]+" "+a[0]);

}

}

图片包含 屏幕截图

描述已自动生成

public class Main {

public static void main(String[] args) {

int k=0;

for(int i=3;i<1000;++i)

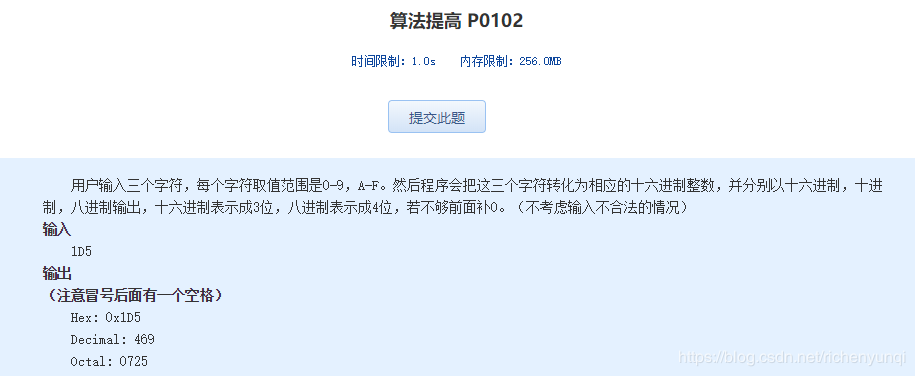
if(i%3==0||i%5==0)

k+=i;

System.out.println(k);

}

}



import java.util.\*;

public class Main {

public static void main(String[] args) {

// TODO Auto-generated method stub

Scanner in = new Scanner(System.in);

String s = in.nextLine();

System.out.println("Hex: 0x" + s);

int k = Integer.parseInt(s, 16);//将16进制字符串转化为10进制整数

System.out.println("Decimal: " + k);

s = Integer.toOctalString(k);//将10进制整数转化为8进制字符串

System.out.print("Octal: ");

for(int i=0;i<4-s.length();++i)

System.out.print("0");

System.out.println(s);

}

}

图片包含 屏幕截图

描述已自动生成

import java.util.\*;

public class Main {

public static int gcd(int a,int b){//欧几里得算法求解最大公约数

return b==0?a:gcd(b,a%b);

}

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

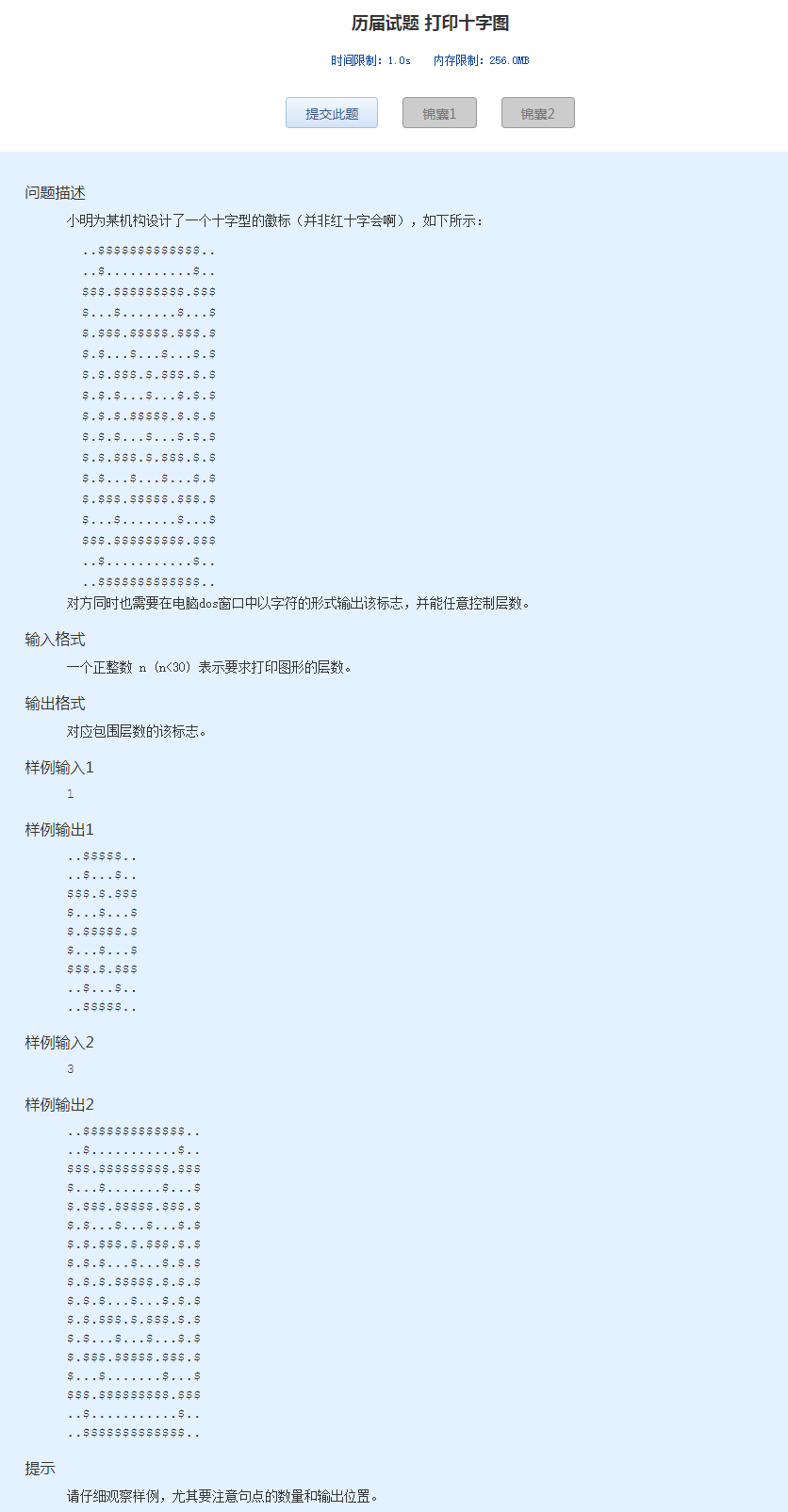
int a=input.nextInt(),b=input.nextInt(),c=input.nextInt();

int k=a/gcd(a,b)\*b;//求出a,b的最小公倍数

System.out.println(k/gcd(k,c)\*c);//输出k,c的最小公倍数

}

}



import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

int N=input.nextInt();

int len=4\*N+5,mid=len/2;//len表示整个图形长宽、mid表示图形中心位置

char [][]A=new char[len][len];//二维数组存储整个图形

for(int i=0;i<len;++i)

for(int j=0;j<len;++j)

A[i][j]='.';//将二维数组所有元素设置为.

for(int i=0;i<5;++i)//填充中间十字架

A[mid][mid-2+i]=A[mid-2+i][mid]='$';

for(int i=1;i<=N;++i){//填充N个十字

for(int j=0;j<4\*i+1;++j){

A[mid+2+2\*i][mid-2\*i+j]=A[mid-2-2\*i][mid-2\*i+j]='$';//填充十字第一行、最后一行

A[mid-2\*i+j][mid+2+2\*i]=A[mid-2\*i+j][mid-2-2\*i]='$';//填充十字第一列、最后一列

}

for(int j=0;j<3;++j){//填充十字拐角

A[mid-2\*i][mid-2\*i-j]=A[mid-2\*i][mid+2\*i+j]='$';

A[mid+2\*i][mid-2\*i-j]=A[mid+2\*i][mid+2\*i+j]='$';

A[mid-2\*i-j][mid-2\*i]=A[mid+2\*i+j][mid-2\*i]='$';

A[mid-2\*i-j][mid+2\*i]=A[mid+2\*i+j][mid+2\*i]='$';

}

}

for(int i=0;i<len;++i){

for(int j=0;j<len;++j)

System.out.print(A[i][j]);

System.out.println();

}

}

}



import java.util.\*;

public class Main {

public static boolean[] visited = new boolean[10];//1~9这9个数字哪些数字已被使用

public static int[] number = new int[9];//储存1~9这9个数字的排列

public static int ans = 0;//ans为最后输出的种类数

public static int compute(int left, int right) {//计算数组number的[left,right]区间所表示的数的值

int a = 0;

for (int i = left; i <= right; ++i)

a = a \* 10 + number[i];

return a;

}

public static void DFS(int index, int N) {//当前需要排列的是number[index]，N为输入的数字

if (index == 9) {////找到了1~9这9个数字的一个排列

for (int i = 0; i < 7; ++i)//分割带分数的整数部分

for (int j = i + 1; j < 8; ++j) {//分割带分数的分子分母

int a = compute(0, i), b = compute(i + 1, j), c = compute(j + 1, 8);//计算带分数的整数、分子、分母的值

if (b % c != 0 || a + b / c != N)//分子不能整除分母或者当前带分数值不等于N

continue;//重新进行循环

++ans;//递增种类数

}

return;//返回上一层

}

for (int i = 1; i < 10; ++i)//遍历1~9这9个数

if (!visited[i]) {//当前遍历到的数还没有使用过

visited[i] = true;//置当前遍历到的数已使用

number[index] = i;//将当前遍历到的数填充到number[index]位置

DFS(index + 1, N);//进行下一层遍历

visited[i] = false;//置当前遍历到的数没使用，遍历下一个数

}

}

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

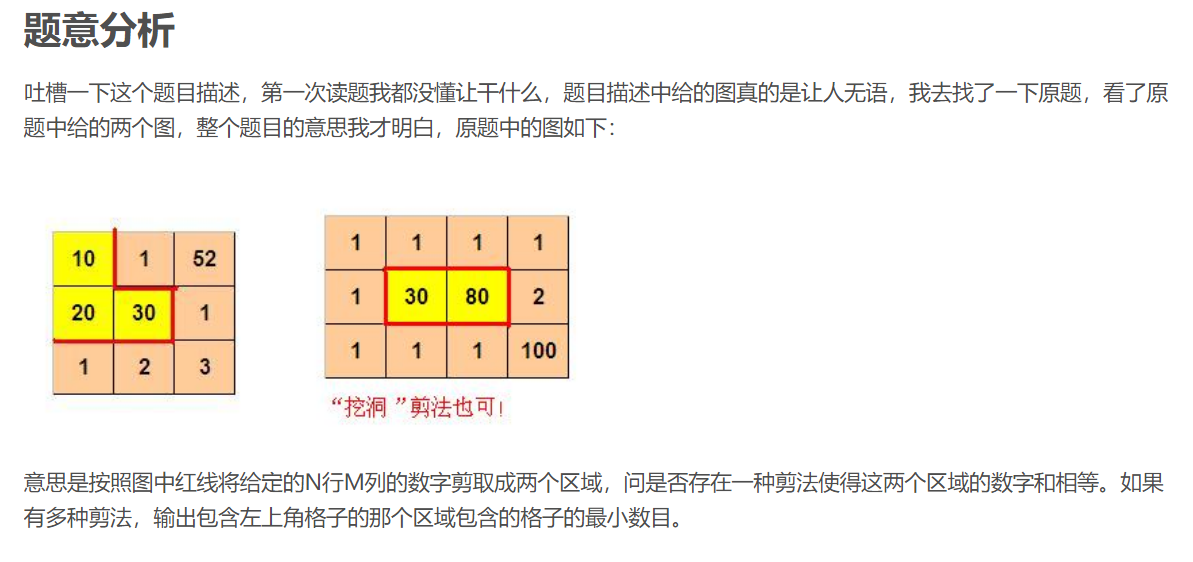
DFS(0,input.nextInt());

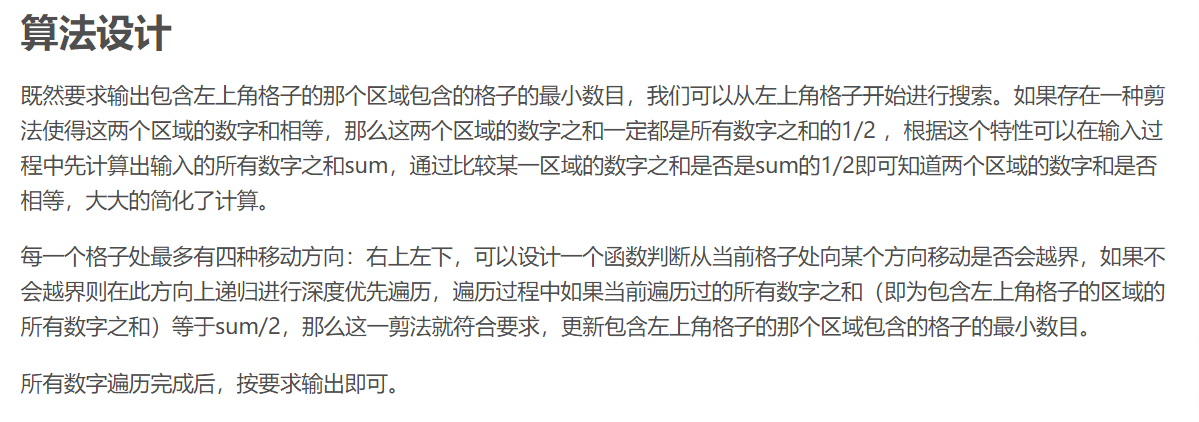
System.out.println(ans);

}

}







import java.util.\*;

public class Main {

// N表示行数，M表示列数，num表示包含左上角格子的那个区域包含的格子的最小数目，sum表示输入的所有数字之和

public static int N, M, num = Integer.MAX\_VALUE, sum = 0;

public static int[][] A = new int[10][10];// 输入的N行M列数字

public static boolean[][] visit = new boolean[10][10];// 表示是否被访问过

public static boolean outOfBorder(int i, int j) {// 判断当前坐标是否越界

if (i < 0 || i >= N || j < 0 || j >= M)

return true;

return false;

}

// 深度优先遍历,i、j表示当前坐标的行列，currentSum表示当前遍历过的数字之和，currentNum表示当前遍历过的数字数量

public static void DFS(int i, int j, int currentSum, int currentNum) {

visit[i][j] = true;// 访问过该坐标下的数字

currentSum += A[i][j];// 更新遍历过的数字之和

++currentNum;// 更新遍历过的数字数量

if (2 \* currentSum >= sum) {// 如果当前遍历过的数字之和大于等于所有数字之和的一半

if (2 \* currentSum == sum)// 如果当前遍历过的数字之和等于所有数字之和的一半

num = Math.min(currentNum, num);// 更新包含左上角格子的那个区域包含的格子的最小数目

visit[i][j] = false;

return;// 回溯到上一结点

}

if (!outOfBorder(i, j + 1) && !visit[i][j + 1])// 向右移动

DFS(i, j + 1, currentSum, currentNum);

if (!outOfBorder(i + 1, j) && !visit[i + 1][j])// 向下移动

DFS(i + 1, j, currentSum, currentNum);

if (!outOfBorder(i, j - 1) && !visit[i][j - 1])// 向左移动

DFS(i, j - 1, currentSum, currentNum);

if (!outOfBorder(i - 1, j) && !visit[i - 1][j])// 向上移动

DFS(i - 1, j, currentSum, currentNum);

visit[i][j] = false;

}

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

M = input.nextInt();

N = input.nextInt();

for (int i = 0; i < N; ++i)

for (int j = 0; j < M; ++j) {

A[i][j] = input.nextInt();

sum += A[i][j];

}

DFS(0, 0, 0, 0);

if (num == Integer.MAX\_VALUE)// num没有变化过，说明没有符合要求的剪法

System.out.println("0");// 输出0

else

System.out.println(num);

}

}



import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

//left、right分别为输入的ID号的最小值和最大值,sum为输入的所有ID号之和，n为重复的ID号

int left=Integer.MAX\_VALUE,right=-1,sum=0,n=0;

HashSet<Integer>s=new HashSet<>();//存储已读取的输入ID号

int N=input.nextInt();//N为输入数据

while(input.hasNext()){//检测数据是否输入完成

N=input.nextInt();

left=Math.min(left,N);//更新最小ID号

right=Math.max(right,N);//更新最大ID号

sum+=N;//更新输入ID号之和

if(s.contains(N))//当前读取的ID号已在s中出现过

n=N;//此ID号即为重复的ID号

else//当前读取的ID号没在s中出现过

s.add(N);//插入到s中

}

System.out.println((left+right)\*(right-left+1)/2-(sum-n)+" "+n);//输出

}

}



import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

int num=0;

char[] s1=input.nextLine().toCharArray(),s2=input.nextLine().toCharArray();

for(int i=0;i<s1.length;++i)

if(s1[i]!=s2[i]){

s1[i]=(s1[i]=='o')?'\*':'o';

s1[i+1]=(s1[i+1]=='o')?'\*':'o';

++num;

}

System.out.println(num);

}

}



import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

int N=input.nextInt(),ans=0;//ans存储补发的糖果数

int []A=new int[N],B=new int[N];//数组A存储小朋友的糖果数，数组B存储小朋友的糖果数的一半

for(int i=0;i<N;++i)

A[i]=input.nextInt();

for(boolean f=true;;f=true){//f为跳出循环的标志

for(int i=0;i<N;++i)

if(A[i]!=A[0])//糖果数不等时

f=false;//置f为false

if(f)//若f仍为true，说明所有糖果数相等

break;//跳出循环

for(int i=0;i<N;++i)//用数组B存储小朋友的糖果数的一半

B[i]=A[i]/2;

for(int i=0;i<N;++i){

A[i]=B[i]+B[(N+i-1)%N];//一轮分糖后，小朋友糖果数

if(A[i]%2==1){//是奇数

++A[i];//小朋友糖果数+1

++ans;//补发糖果

}

}

}

System.out.println(ans);

}

}



import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

int[][] direction = { // 上右下左四个方向

{ -1, 0 }, { 0, 1 }, { 1, 0 }, { 0, -1 } };

int directIndex = -1, N = input.nextInt(), M = input.nextInt();// 当前方向,N行M列

int[][] A = new int[N][M];// 读取的地图矩阵

for (int i = 0; i < N; ++i)

for (int j = 0; j < M; ++j)

A[i][j] = input.nextInt();

int a = input.nextInt(), b = input.nextInt();

char c = input.next().charAt(0);

int K = input.nextInt();

// 确定初始方向

if (c == 'U')

directIndex = 4 \* K;

else if (c == 'D')

directIndex = 4 \* K + 2;

else if (c == 'L')

directIndex = 4 \* K + 3;

else if (c == 'R')

directIndex = 4 \* K + 1;

for (int ii = 0; ii < K; ++ii) {// K步模拟

if (A[a][b] == 1) {// 如果是黑格

++directIndex;// 右转90度

A[a][b] = 0;// 将当前格子置为白格

} else if (A[a][b] == 0) {// 如果是白格

--directIndex;// 左转90度

A[a][b] = 1;// 将当前格子置为黑格

}

a += direction[directIndex % 4][0];// 前移一格

b += direction[directIndex % 4][1];

}

System.out.println(a + " " + b);// 输出当前位置

}

}