1 [非利克瑞尔数](http://59.67.225.28/admin/courseAdmin/assignmentAdmin/javascript:displayProContent('problemDesc4550',%20'4550');)  
利克瑞尔数（Lychrel Number）指的是将该数各数位逆序翻转后形成的新数相加，并将该过程反复迭代后，结果永远无法是一个回文数的自然数。57就是一个非利克瑞尔数：57+75=132, 132+231=363，363是一个回文数。请编写程序，输入一个自然数（非利克瑞尔数），请计算其最终的回文数是多少及每次迭代过程。注意：假设输入的整数和中间产生的整数都不超过int数据类型的表示范围。

package p15881;

import java.util.Scanner;

public class P15881 {

public static void main(String[] args) {

// TODO Auto-generated method stub

int m=0,n=0,t=0;

Scanner sc=new Scanner(System.in);

n=sc.nextInt();

sc.close();

m=reverse(n);

while(m!=n){

System.out.println((t+1)+":"+n+"+"+m+"="+(m+n));

n=m+n;

m=reverse(n);

t++;

}

if(m==n){

System.out.println(m);

}

}

public static int reverse(int n){

int m=0;

m=0;

while(n>0){

m=m\*10+n%10;

n/=10;

}

return m;

}

}

**2** 合数分解

由数学基本定理可知：任何一个大于1的非素数整数（即合数）都可以唯一分解成若干个素数的乘积。编写程序，从控制台读入一个合数（合数的大小不会超过int数据类型表示的范围），求该合数分解成素数的最小集。该最小素数集由合数分解成的素数组成，并且重复的素数只保留一个。按从小到大的顺序输出求得的最小素数集。

package p15940;

import java.util.Scanner;

public class P15940 {

public static void main(String[] args) {

// TODO Auto-generated method stub

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

for(int i=2;n>1;i++){

if(n%i==0){

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

while(n%i==0){

n/=i;

}

}

}

sc.close();

}

}

3 **工资**

假设税前工资和税率如下（s代表税前工资，t代表税率）：  
s<1000    t=0%  
1000<=s<2000      t=10%  
2000<=s<3000  t=15%  
3000<=s<4000     t=20%  
4000<=s             t=25%  
编写一程序，要求用户输入税前工资额，然后用switch语句计算税后工资额。

package Test;

import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

import java.io.InputStream;

import java.io.OutputStream;

import java.io.OutputStreamWriter;

import java.math.BigDecimal;

import java.util.Scanner;

public class Demo {

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

Scanner sc = new Scanner(System.in);

float salary;

double tax;

salary = sc.nextFloat();

switch(((int)salary) / 1000)

{

case 0:

tax = 1.0;

break;

case 1:

tax = 0.9;

break;

case 2:

tax = 0.85;

break;

case 3:

tax = 0.8;

break;

default:

tax = 0.75;

}

System.out.printf("%.2f", salary \* tax);

}

}

**4 找最大最小整数**

【问题描述】编写一个程序，用户输入若干整数，试找出其中的最大数和最小数。  
【输入形式】用户在第一行待输入数据个数，在第二行输入数据。  
【输出形式】程序在下一行输出数据的最大值和最小值

package p16121;

import java.util.Scanner;

public class P16121 {

public static void main(String[] args) {

// TODO Auto-generated method stub

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

int min=0,max=0,temp;

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

temp=sc.nextInt();

if(i==0){

min=temp;

max=temp;

}

else{

if(temp<min){

min=temp;

}

if(temp>max){

max=temp;

}

}

}

System.out.println(max+""+min);

sc.close();

}

}

5**各位数字和**

【问题描述】编写函数int sum(int x)，求整数x的各位数字之和。在main函数中测试该函数：从键盘输入一非负整数，然后调用sum函数计算各位数字之和并输出结果。   
【输入形式】输入一个正整数。   
【输出形式】输出该整数各位数字之和。

package Test;

import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

import java.io.InputStream;

import java.io.OutputStream;

import java.io.OutputStreamWriter;

import java.math.BigDecimal;

import java.util.Scanner;

public class Demo {

public static int sums(int num)

{

int sum;

sum = 0;

while (num != 0)

{

sum += num % 10;

num /= 10;

}

return sum;

}

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

int number;

Scanner sc = new Scanner(System.in);

number = sc.nextInt();

System.out.printf("%d", sums(number));

}

}

**6 求三角形面积**

【问题描述】若已知三角形三个边的长度分别为a,b,c（并假设三个边长度的单位一致，在本编程题中忽略其单位），则可以利用公式S=s(s-a)(s-b)(s-c)求得三角形的面积，其中：s=(a+b+c)/2。编程实现从控制台读入以整数表示的三个边的长度（假设输入的长度肯定可以形成三角形），然后利用上述公式计算面积并输出，结果小数点后保留3位有效数字。

【输入形式】从控制台输入三个整数表示三角形三个边的长度，以空格分隔三个整数。

【输出形式】向控制台输出求得的三角形的面积，小数点后保留三位有效数字。package p15948;

import java.util.Scanner;

public class P15948 {

public static void main(String[] args) {

// TODO Auto-generated method stub

Scanner sc=new Scanner(System.in);

double a=sc.nextDouble(),b=sc.nextDouble(),c=sc.nextDouble();

double S=0,s=(a+b+c)/2;

S=s\*(s-a)\*(s-b)\*(s-c);

System.out.println(String.format("%.3f",Math.sqrt(S)));

sc.close();

}

}

**7 求最大公因子**

【问题描述】用递归方法编写求最大公因子程序。两个正整数x和y的最大公因子定义为：如果y<=x且x mod y＝0时，gcd(x,y)=y;如果y>x时，gcd(x,y)=gcd(y,x);其他情况，gcd(x,y)=gcd(y,x mod y)

【输入形式】用户在第一行输入两个数字，数字之间用空格分割。

【输出形式】程序在下一行输出前面输入的两个数字的最大公因子。

package p23943;

import java.util.Scanner;

public class P23943 {

public static void main(String[] args) {

// TODO Auto-generated method stub

Scanner sc=new Scanner(System.in);

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

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

sc.close();

}

public static int gcd(int a,int b){

int x,y;

x=a>b?a:b;

y=a<b?a:b;

if(x==y||x%y==0){

return y;

}

else{

int mark=0;

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

if(x%i==0&&y%i==0){

mark=i;

}

}

return mark;

}

}

}

**8 人民币兑换**

输入一个人民币的整数值（100以内以元为单位），编程找到用10元、5元、2元、1元表示的总数量的最小组合方式。从控制台输入一个整数值，表示以元为单位的人民币币值。【输出形式】向控制台输出四个整数（以空格分隔），分别表示兑换成的10元、5元、2元、1元人民币的数量，若没有某个币值，则对应输出0。

package p23892;

import java.util.Scanner;

public class P23892 {

public static void main(String[] args) {

// TODO Auto-generated method stub

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

int a=0,b=0,c=0,e=0;

a=n/10;

n=n-a\*10;

b=n/5;

n=n-b\*5;

c=n/2;

n=n-c\*2;

e=n/1;

System.out.println(a+""+b+""+c+""+e);

sc.close();

}

}

**9 摄氏华氏温度转换**

【问题描述】假如用C表示摄氏温度，F表示华氏温度，则有：F=C\*9/5+32。输入一整数表示摄氏温度，根据该公式编程求对应的华氏温度，结果小数点后保留一位有效数字。

【输入形式】从控制台读入一个整数，表示摄氏温度。

【输出形式】向控制台输出转换后的华氏温度，结果小数点后保留一位有效数字。

package p23891;

import java.util.Scanner;

public class P23891 {

public static void main(String[] args) {

// TODO Auto-generated method stub

Scanner sc=new Scanner(System.in);

int c=sc.nextInt();

float f=(float)c\*9/5+(float)32;

System.out.println(String.format("%.1f", f));

sc.close();

}

}

**1 判断某日期为该年中的第几天**

【问题描述】判断某日期为该年中的第几天

【输入形式】分别输入表示日期的年、月、日3个整数，以空格隔开  
【输出形式】输出此日期在本年中为第几天的天数

package test2;

import java.util.Scanner;

public class Ex210 {

public static void main(String[] args){

Scanner scn = new Scanner(System.in);

int year = scn.nextInt();

int month = scn.nextInt();

int day = scn.nextInt();

int days=0;

for(int m=1; m<month; m++){

if(m==1||m==3||m==5||m==7||m==8||m==10||m==12){

days+=31;

}else if(m==4||m==6||m==9||m==11){

days+=30;

}

}

if(month>2){

if(year%4==0 && year%100!=0 || year%400==0){

days+=29;

}else{

days+=28;

}

}

System.out.println(days+day);

}

}

**2 最大公约数和最小公倍数**

【问题描述】输入两个正整数a和b（0&le;a，b&le;1000000），求出其最大公约数和最小公倍数并输出。  
【输入文件】从标准输入读取一行，是两个整数a和b，以空格分隔。  
【输出文件】向标准输出打印以空格分隔的两个整数，分别是a、b的最大公约数和最小公倍数。在输出末尾要有一个回车符。

package Test;

import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

import java.io.InputStream;

import java.io.OutputStream;

import java.io.OutputStreamWriter;

import java.math.BigDecimal;

import java.util.Scanner;

public class Demo {

public static void main(String[] args) {

long a, b;

long r;

long m, n;

Scanner sc = new Scanner(System.in);

a=sc.nextLong();

b=sc.nextLong();

if (a > b) {

r = a;

a = b;

b = r;

}

m = a; n = b;

while (m!=0) {

r = n % m;

n = m;

m = r;

}

System.out.printf("%d %d\n", n, b / n \* a);

}

}

3 **素数判断**

【问题描述】编写一个函数isprime(n)。判断整数n是否为素数。编写程序使用此函数，当输入一个整数时，对它进行判断，当为素数时，输出1。否则，输出0。

【输入形式】控制台输入一个整数。

【输出形式】控制台输出判断结果0或者1。

package p16208;

import java.util.Scanner;

public class P16208 {

public static void main(String[] args) {

// TODO Auto-generated method stub

Scanner sc=new Scanner(System.in);

System.out.println(isprime(sc.nextInt()));

sc.close();

}

public static int isprime(int n){

if(n==2){

return 1;

}

for(int i=2;i<(int)(Math.sqrt(n))+1;i++){

if(n%i==0){

return 0;

}

}

return 1;

}

}

4**根据输入的运算符组织运算**

【问题描述】根据输入的运算符对操作数进行计算

【输入形式】 输入1个操作符（+  -  \*  /  四个中的一个）和2个操作数  
【输出形式】 当操作符输入正确的时候，输出这2个操作数进行此操作符对应的运算后的结果； 若操作符输入错误，则输出字符串error

package test2;

import java.util.Scanner;

public class Ex27 {

public static void main(String[] args) {

Scanner scn = new Scanner(System.in);

char operator=scn.next().charAt(0);

double x=scn.nextDouble();

double y=scn.nextDouble();

if(operator=='+'){

System.out.println(""+x+operator+y+"="+(x+y));

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

System.out.println(""+x+operator+y+"="+(x-y));

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

System.out.println(""+x+operator+y+"="+(x\*y));

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

System.out.println(""+x+operator+y+"="+(x/y));

}else{

System.out.println("error");

}

}

}

**5 数据加密**

【问题描述】 某公司传递数据时采用加密方式，数据为四位整数，加密规则如下： 每位数字都加上5，用和除以10的余数代替该数字，再将第一位和第四位交换，第二位和第三位交换  
【输入形式】 四位整数  
【输出形式】 经过加密处理后的四位整数

package test2;

import java.util.Scanner;

public class Excercise10 {

public static void main(String[] args) {

Scanner scn = new Scanner(System.in);

int number = scn.nextInt();

int a= (number%10+5)%10;

number/=10;

int b=(number%10+5)%10;

number/=10;

int c=(number%10+5)%10;

number/=10;

int d=(number%10+5)%10;

int t=a;

a=d;

d=t;

t=b;

b=c;

c=t;

System.out.println(""+d+c+b+a);

}

}

|  |
| --- |
|  |

|  |
| --- |
| **6 相亲数b**  【问题描述】2500年前数学大师毕达哥拉斯就发现，220和284两数之间存在着奇妙的联系： 220的因数之和（除了自身之外的因数）为：110+55+44+22+20+11+10+5+4+2+1=284，284的因数之和为：142+71+4+2+1=220。 毕达哥拉斯把这样的数对称为相亲数。输入两个正整数（大于1并且在int的表示范围之内），分别计算它们所有因数之和（除了自身之外的因数），并判断它们是否是一对相亲数。  【输入形式】从标准输入输入两个正整数，以一个空格分隔这两个正整数。  【输出形式】以输入的先后顺序分行输出：输入的正整数，后跟英文逗号&ldquo;,&rdquo;，再从除自身之外的最大的因数开始输出其因数相加的公式（最小的因数1之后没有加号），最后输出英文等号&ldquo;=&rdquo;和因数之和。注意：所有输出元素间无空格。 若它们是一对相亲数，则在新的一行上输出1；若不是则在新的一行上输出0。 |

package Test;

import java.text.NumberFormat;

import java.util.ArrayList;

import java.util.HashSet;

import java.util.List;

import java.util.Scanner;

import java.util.Set;

import java.math.\*;

public class Demo {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int a,b;

a=sc.nextInt();

b=sc.nextInt();

List<Integer> l1 = new ArrayList<Integer>();

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

if(a%i==0) {

l1.add(i);

}

}

List<Integer> l2 = new ArrayList<Integer>();

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

if(b%i==0) {

l2.add(i);

}

}

int sum1 = 0, sum2 = 0;

System.out.print(a+",");

for(int i=0;i<l1.size();i++) {

if(i != 0)

System.out.print("+");

System.out.print(l1.get(l1.size()-1-i));

sum1 += l1.get(l1.size()-1-i);

}

System.out.println("="+sum1);

System.out.print(b+",");

for(int i=0;i<l2.size();i++) {

if(i != 0)

System.out.print("+");

System.out.print(l2.get(l2.size()-1-i));

sum2 += l2.get(l2.size()-1-i);

}

System.out.println("="+sum2);

if(sum1 == b && sum2 == a) {

System.out.println("1");

} else {

System.out.println("0");

}

}

}

package p15905;

import java.util.Scanner;

public class P15905 {

public static void main(String[] args) {

// TODO Auto-generated method stub

int i=0,m1=0,m2=0,sum1=0,sum2=0;

Scanner sc=new Scanner(System.in);

m1=sc.nextInt();

m2=sc.nextInt();

sum1=0;

System.out.print(m1+",");

for(i=m1-1;i>0;i--){

if(m1%i==0){

if(sum1==0){

System.out.print(i);

}

else{

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

}

sum1+=i;

}

}

System.out.println("="+sum1);

sum2=0;

System.out.print(m2+",");

for(i=m2-1;i>0;i--){

if(m2%i==0){

if(sum2==0){

System.out.print(i);

}

else{

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

}

sum2+=i;

}

}

System.out.println("="+sum2);

if((sum1==m2)&&(sum2==m1)){

System.out.println(1);

}

else{

System.out.println(0);

}

sc.close();

}

}

**7 北京地铁计价程序**

【问题描述】北京地铁按公里计价的规则为：6km（含）内为3元；6~12km（含）为4元；12~22km（含）为5元；22~32km（含）为6元；32km以上每加1元可乘坐20km.

【输入形式】输入要乘坐地铁的距离（整数），以km为单位

【输出形式】输出乘坐这段距离地铁所需要的费用，以元为单位

package test2;

import java.util.Scanner;

public class Excercise8 {

public static void main(String[] args) {

//System.out.print("input distance:");

Scanner scn = new Scanner(System.in);

int distance = scn.nextInt();

int price;

if(distance<=6&& distance>0){

price=3;

}else if(distance<=12){

price=4;

}else if(distance<=22){

price=5;

}else if(distance<=32){

price=6;

}else {

if((distance-32)%20!=0){

price = 6+ (distance-32)/20+1;

}else{

price = 6+ (distance-32)/20;

}

}

System.out.printf("price=%d\n",price);

}

}

8完全数

【问题描述】一个整数，如果其所有小于它本身的因子（包括1）之和正好等于该数，则称其为&ldquo;完全数&rdquo;。编写程序计算某一范围内的所有&ldquo;完全数&rdquo;。

【输入形式】从控制台输入数据范围的下限m和上限n，m和n都为整数，且1<=m<=n<=10000，m和n之间用一个空格分割。

【输出形式】向控制台输出m和n之间的所有&ldquo;完全数&rdquo;（包括m和n本身），所有输出的数据以从小到大的顺序显示在一行上，各数之间以一个空格分割。若该范围内没有完全数，则输出字符串No Answer。

package p16108;

import java.util.Scanner;

public class P16108 {

public static void main(String[] args) {

// TODO Auto-generated method stub

Scanner sc=new Scanner(System.in);

int start=sc.nextInt();

int end=sc.nextInt();

int has=0;

for(int i=start;i<=end;i++){

if(judge(i)==1){

has++;

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

}

}

if(has==0){

System.out.println("No Answer");

}

sc.close();

}

public static int judge(int n){

if(n==1){

return 0;

}

int result=1;

int n\_sqrt=(int)(Math.sqrt(n))+1;

int i=2;

while(i<n\_sqrt){

if(n%i==0){

if(i==(int)(n/i)){

result+=i;

}

else{

result+=i+(int)(n/i);

}

i++;

}

else{

i++;

}

}

if(n==result){

return 1;

}

else{

return 0;

}

}

}

9**对数据分页**

【问题描述】设有若干行数据（rows）需要显示，且已规定每页显示数目（pageSize），计算这些数据需要分为多少页显示。  
【输入形式】分别输入要查询的数据（rows）和每页显示数目（pageSize），以回车符隔开。

【输出形式】输出显示这些数据所需要的页数

package test2;

import java.util.Scanner;

public class Excercise7 {

public static void main(String[] args) {

Scanner console = new Scanner(System.in);

int rows = console.nextInt();

int size = console.nextInt();

int pages = pages(rows, size);

System.out.println(""+pages);

}

public static int pages(int rows, int size){

int p = rows%size==0 ? rows/size: rows/size+1;

return p;

}

}

10 **求水仙花数**

【问题描述】编写一个程序，输入一个正整数N(N大于等于100小于等于999)，求出100~N之间的所有水仙花数。所谓的水仙花数是：如果一个三位数的个位数、十位数、百位数的立方和等于该数自身，就称这个数为水仙花数。  
【输入形式】输入一个正整数N。  
【输出形式】输出从100到N之间的所有水仙花数，每个数以回车结束。

package Test;

import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

import java.io.InputStream;

import java.io.OutputStream;

import java.io.OutputStreamWriter;

import java.math.BigDecimal;

import java.util.Scanner;

public class Demo {

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

int N,i;

int a,b,c;

Scanner sc = new Scanner(System.in);

N = sc.nextInt();

if(N < 100 || N > 999) {

}

for( i = 100 ; i <= N; i++)

{

a = i / 100 ;

b = (i / 10)%10;

c =( i % 100) %10;

if( a\*a\*a+b\*b\*b+c\*c\*c == i)

System.out.printf("%d\n",i);

}

}

}

**1 二维整型数组的“最小点”**

【问题描述】求二维整型数组的”最小点”。二维数组的”最小点”定义为：某个数是所在行的最小值，并且是所在列的最小值。注意：某行或某列上可能有多个”最小点”。

【输入形式】从控制台读入二维数组。第一行只有以空格分隔的两个正整数n和m（n,m<=10），n代表二维数组的行数，m代表二维数组的列数。然后在后续n行上输入二维数组的元素，每行有m个以若干空格分隔的整数，代表二维数组在该行上的所有元素。

【输出形式】向控制台输出二维数组的”最小点”，按行下标、列下标从小到大的顺序输出，每行一个，先输出”最小点”数值，再输出对应的行数、列数（行列都从1开始计数），以一个空格分隔。

package p16166;

import java.util.ArrayList;

import java.util.Scanner;

public class P16166 {

@SuppressWarnings("resource")

public static void main(String[] args) {

// TODO Auto-generated method stub

Scanner sc=new Scanner(System.in);

int a=sc.nextInt();

int b=sc.nextInt();

int [][]matrix=new int[a][b];

int []smallest=new int[a];

int small=0;

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

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

matrix[i][j]=sc.nextInt();

}

small=matrix[i][0];

for(int m=0;m<b;m++){

if(matrix[i][m]<small){

small=matrix[i][m];

}

}

smallest[i]=small;

}

int hassmaller=0;

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

ArrayList<String> mark=new ArrayList<>();

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

if(smallest[i]==matrix[i][n]){

mark.add(String.valueOf(n));

}

}

for(int x=0;x<mark.size();x++){

hassmaller=0;

for(int p=0;p<a;p++){

if(matrix[p][Integer.parseInt(mark.get(x))]<smallest[i]){

hassmaller=1;

break;

}

}

if(hassmaller==0){

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

System.out.print(i+1+"");

System.out.println(Integer.parseInt(mark.get(x))+1);

}

}

}

}

}

2 **合并整数**

【问题描述】输入两组正整数，每组整数不超过20个，并且每组整数中不存在重复数据，也不存在是另一个数据整数倍的数据。编写一个程序按下面要求合并两组整数并输出：1. 合并时如果一个数据是另一个数据的整数倍，则剔除；若数据相同，则只保留一个；2. 从小到大顺序输出合并后数据。

【输入形式】先从控制台输入第一组整数的个数，然后在下一行输入第一组整数，各整数之间以一个空格分隔；然后按照同样的方式输入第二组整数。【输出形式】在标准输出上按从小到大顺序输出合并后的数据，各整数之间以一个空格分隔，最后一个整数后也可以有一个空格。

package p15947;

import java.util.Scanner;

public class P15947 {

public static void main(String[] args) {

// TODO Auto-generated method stub

Scanner sc=new Scanner(System.in);

int n1=sc.nextInt();

int []num1=new int[n1];

for(int i=0;i<n1;i++){

num1[i]=sc.nextInt();

}

int n2=sc.nextInt();

int []num2=new int[n2];

for(int i=0;i<n2;i++){

num2[i]=sc.nextInt();

}

int []num=new int[n1+n2];

int isdouble=0,k=0,appear=0;

for(int i=0;i<n1;i++){

isdouble=0;

for(int j=0;j<n2;j++){

if(num1[i]%num2[j]==0){

if(num1[i]==num2[j]){

continue;

}

isdouble=1;

break;

}

}

if(isdouble==0){

appear=0;

for(int t=0;t<k;t++){

if(num1[i]==num[t]){

appear=1;

break;

}

}

if(appear==0){

num[k]=num1[i];

k++;

}

}

}

for(int i=0;i<n2;i++){

isdouble=0;

for(int j=0;j<n1;j++){

if(num2[i]%num1[j]==0){

if(num2[i]==num1[j]){

continue;

}

isdouble=1;

break;

}

}

if(isdouble==0){

appear=0;

for(int t=0;t<k;t++){

if(num2[i]==num[t]){

appear=1;

break;

}

}

if(appear==0){

num[k]=num2[i];

k++;

}

}

}

int tmp=0;

for(int p=0;p<k;p++){

for(int q=0;q<k-1;q++){

if(num[q]>num[q+1]){

tmp=num[q];

num[q]=num[q+1];

num[q+1]=tmp;

}

}

}

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

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

}

sc.close();

}

}

**3 对称矩阵乘法运算1**

【问题描述】对称矩阵有性质a[i][j] = a[j][i]。为了节省空间，可以只存储对称矩阵的下三角元素。如下面左边的对称矩阵可以以"行"为序存储成右边的一维形式。

1 2 3 4  
2 5 7 9                                                1 2 5 3 7 8 4 9 -1 0  
3 7 8 -1  
4 9 -1 0

编写程序，实现以一维形式输入的对称矩阵的乘法，对称矩阵最大为9\*9。

【输入形式】先从控制台输入对称矩阵的阶数和一维形式的数据个数（以一个空格分隔），然后以一维形式分行输入两个对称矩阵的数据，数据之间以一个空格分隔。

【输出形式】在标准输出上分行输出两对称矩阵的乘积，每行上的数据以一个空格分隔，每行最后一个数据后要有一个空格。

package p15973;

import java.util.Scanner;

public class P15973 {

public static void main(String[] args) {

// TODO Auto-generated method stub

Scanner sc=new Scanner(System.in);

int [][]a=new int[9][9];

int [][]b=new int[9][9];

int [][]c=new int[9][9];

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

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

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

a[i][j]=sc.nextInt();

a[j][i]=a[i][j];

}

}

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

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

b[i][j]=sc.nextInt();

b[j][i]=b[i][j];

}

}

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

for(int k=0;k<m;k++){

c[i][k]=0;

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

c[i][k]+=a[i][j]\*b[j][k];

}

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

}

System.out.println();

}

sc.close();

}

}

4 判断一个矩阵是另一个矩阵的子矩阵

【问题描述】从标准输入中输入一个N（N<=9）阶矩阵和一个M（M<=N）阶矩阵，判断矩阵M是否是N的子矩阵，若是则输出M在N中的起始位置，若不是则输出-1。若矩阵M能与N中某一区域完全相等，则称M是N的子矩阵。

【输入形式】从标准输入读取矩阵。

第一行只有一个整数N，代表第一个矩阵的阶数。后续有N行输入，每行有N个以若干空格分隔的整数，代表该矩阵在该行上的所有元素。

输入完N阶矩阵后，再在下一行输入一个整数M，代表第二个矩阵的阶数。后续有M行输入，每行有M个以若干空格分隔的整数，代表该矩阵在该行上的所有元素。

【输出形式】输出M在N中的起始位置，即N中的第几行第几列，两个数字用逗号&ldquo;,&rdquo;分隔（从第1行第1列开始计数，即：矩阵第一个元素的位置为：1,1。

若N有多个子矩阵与M矩阵完全相同，则输出首先找到的起始位置，即行最小的位置，若行相同，则为列最小的位置。

若M不是N的子矩阵，则输出-1。

package p16241;

import java.util.Scanner;

public class P16241 {

@SuppressWarnings("resource")

public static void main(String[] args) {

Scanner sc=new Scanner(System.in);

int N=sc.nextInt();

int [][]x=new int[N][N];

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

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

x[i][j]=sc.nextInt();

}

}

int M=sc.nextInt();

int [][]y=new int[M][M];

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

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

y[i][j]=sc.nextInt();

}

}

for(int p=0;p<N-M+1;p++){

for(int q=0;q<N-M+1;q++){

if(Issame(p,q,M,x,y)==1){

System.out.print((p+1)+",");

System.out.println(q+1);

System.exit(0);

}

}

}

System.out.println(-1);

}

public static int Issame(int p,int q,int M,int x[][],int y[][]){

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

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

if(x[p+m][q+n]!=y[m][n]){

return 0;

}

}

}

return 1;

}

}

5 大奖赛评分A

【问题描述】当前许多歌手大奖赛评分时，为了体现公平，在评委给出分数后统计平均得分时，都会去掉最高分和最低分。编写程序，读入评委打分（分数都是整数，评委人数大于等于5，小于等于50），去掉两个最高分和两个最低分，计算并输出平均得分（小数点后保留两位有效数字）。

【输入形式】从控制台输入评委人数，然后在下一行输入评委的打分（以一个空格分隔的整数）。

【输出形式】在标准输出上输出平均得分（小数点后保留两位有效数字）。

package p16002;

import java.text.DecimalFormat;

import java.util.Scanner;

public class P16002 {

public static void main(String[] args) {

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

int []marklist=new int[n];

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

marklist[i]=sc.nextInt();

}

int temp=0;

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

for(int q=0;q<n-1;q++){

if(marklist[q]>marklist[q+1]){

temp=marklist[q];

marklist[q]=marklist[q+1];

marklist[q+1]=temp;

}

}

}

double finalmark=0;

temp=0;

for(int i=2;i<n-2;i++){

temp+=marklist[i];

}

finalmark=(double)temp/(double)(n-4);

DecimalFormat gs=new DecimalFormat("0.00");

System.out.println(gs.format(finalmark));

sc.close();

}

}

6区间合并

【问题描述】从标准输入读入n（大于等于2，小于等于100）个闭区间[ai,bi]（ai小于bi，且ai,bi均为int范围内的整数），编写程序将这些区间合并为不相交的闭区间。例如：区间[1,2]、[2,5]、[3,8]、[9,12]可以合并为区间[1,8]和[9,12]。将合并后的闭区间按照升序排列输出。

【输入形式】先从标准输入读入区间个数n，然后从下一行开始分行读入各个闭区间，每个闭区间包含由一个空格分隔的整数ai和bi。

【输出形式】按照升序分行输出合并后的闭区间[xi,yi]，xi和yi之间以一个空格分隔。

package p15890;

import java.util.Scanner;

public class P15890 {

public static void main(String[] args) {

Scanner sc=new Scanner(System.in);

int [][]inter1=new int[100][2];

int [][]inter2=new int[100][2];

int num=0,i=0,j=0,m=0,n=0,k=0,found=0,min=0;

num=sc.nextInt();

for(i=0;i<num;i++){

inter1[i][0]=sc.nextInt();

inter1[i][1]=sc.nextInt();

}

for(i=0;i<num;i++)

{

min=i;

for(j=i+1;j<num;j++)

{

if(inter1[j][0]<inter1[min][0])

min=j;

}

if(min!=i)

{

m=inter1[min][0];

n=inter1[min][1];

inter1[min][0]=inter1[i][0];

inter1[min][1]=inter1[i][1];

inter1[i][0]=m;

inter1[i][1]=n;

}

}

k=0;

for(i=0;i<num;i++)

{

m=inter1[i][0];

n=inter1[i][1];

found=0;

for(j=0;j<k;j++)

{

if(m>inter2[j][1] || n<inter2[j][0])

continue;

if(m>=inter2[j][0])

{

if(n>inter2[j][1])

inter2[j][1]=n;

found=1;

break;

}

if(n<=inter2[j][1])

{

if(m<inter2[j][0])

inter2[j][0]=m;

found=1;

break;

}

}

if(found==0)

{

inter2[k][0]=m;

inter2[k][1]=n;

k++;

}

}

for(i=0;i<k;i++)

System.out.println(inter2[i][0]+" "+inter2[i][1]);

sc.close();

}

}

7 矩阵替换B

【问题描述】先输入两个矩阵A和B，然后输入替换位置（右下角），编写程序将矩阵A中以替换位置为右下角的子矩阵（与B同样大小）替换为B，并输出替换后的矩阵。

【输入形式】从控制台先输入矩阵A的行数和列数（行数和列数均大于等于1，小于等于20），然后在新的行上输入矩阵A的各行数字（以一个空格分隔的整数）。再以同样的方式输入矩阵B。最后输入替换位置（用一个空格分隔的两个整数表示，行数和列数都从1开始计数，因此两个整数都大于等于1）。

【输出形式】在标准输出上分行输出替换后的矩阵，每行中各数字之间以一个空格分隔。

package p15998;

import java.util.Scanner;

public class P15998 {

public static void main(String[] args) {

Scanner sc=new Scanner(System.in);

int m1=sc.nextInt(),n1=sc.nextInt();

int [][]a=new int[20][20];

for(int i=0;i<m1;i++){

for(int j=0;j<n1;j++){

a[i][j]=sc.nextInt();

}

}

int m2=sc.nextInt(),n2=sc.nextInt();

int [][]b=new int[20][20];

for(int i=0;i<m2;i++){

for(int j=0;j<n2;j++){

b[i][j]=sc.nextInt();

}

}

int m3=sc.nextInt(),n3=sc.nextInt();

int [][]c=new int[20][20];

for(int i=0;i<m1;i++){

for(int j=0;j<n1;j++){

if((i<m3-m2)||(i>m3-1)){

c[i][j]=a[i][j];

}

else if((j<n3-n2)||(j>n3-1)){

c[i][j]=a[i][j];

}

else{

c[i][j]=b[i-m3+m2][j-n3+n2];

}

}

}

for(int i=0;i<m1;i++){

for(int j=0;j<n1;j++){

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

}

System.out.println();

}

sc.close();

}

}

8 最小素数集

【问题描述】由数学基本定理可知：任何一个大于1的非素数整数都可以唯一分解成若干个素数的乘积。编写程序，从控制台读入一组大于1的整数（小于等于20个，且每个整数的大小不会超过int数据类型表示的范围），求这些整数分解成素数的最小集。该最小素数集是所有整数分解成的素数的并集（若输入的整数是素数，则该素数可以直接加入最小素数集），并且重复的素数只保留一个。按从小到大的顺序输出求得的最小素数集。

【输入形式】先从控制台输入整数的个数，然后在下一行输入所有整数，各整数之间以一个空格分隔。

【输出形式】在标准输出上按从小到大顺序输出求得的最小素数集，各素数之间以一个空格分隔，最后一个整数后也可以有一个空格。

package p15911;

import java.util.Scanner;

public class P15911 {

public static void main(String[] args) {

// TODO Auto-generated method stub

Scanner sc=new Scanner(System.in);

int i=0,j=0,k=0,m=0,n=0,p=0,num=0;

int []primes=new int[50];

int []results=new int[50];

n=sc.nextInt();

m=0;

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

num=sc.nextInt();

j=0;

if(isprime(num)==1){

primes[j++]=num;

}

else{

k=2;

while(num>1){

if(num%k==0){

primes[j++]=k;

while(num%k==0){

num/=k;

}

}

k++;

}

}

for(k=0;k<j;k++){

for(p=0;p<m;p++){

if(primes[k]==results[p]){

break;

}

}

if(p==m){

results[m++]=primes[k];

}

}

}

for(i=0;i<m;i++){

n=i;

for(j=i+1;j<m;j++){

if(results[j]<results[n]){

n=j;

}

}

if(n!=i){

num=results[i];

results[i]=results[n];

results[n]=num;

}

}

for(i=0;i<m;i++){

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

}

sc.close();

}

public static int isprime(int n){

int i=0,m=0;

m=(int)Math.sqrt(n);

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

if(n%i==0){

return 0;

}

}

return 1;

}

}

9 学生成绩排序

【问题描述】对某班学生成绩排序。从键盘依次输入某班学生的姓名和成绩（一个班级人数最多不超过50人）并保存，然后分别按学生成绩由高到低顺序输出学生姓名和成绩，成绩相同时，则按输入次序排序。  
【输入形式】从键盘依次输入最多不超过50个学生的学生姓名和成绩：第一行输入班级学生人数；在单独行上输入空格隔开的学生姓名和成绩，其中学生成绩是整数。  
【输出形式】按学生成绩由高到低顺序输出学生姓名和成绩，每行输出一位学生的姓名和成绩，其中姓名（英文）占15位，成绩占5位，均按缺省方式对齐。成绩相同时按输入次序排序。

package p16124;

import java.util.Scanner;

public class P16124 {

public static void main(String[] args) {

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

String []name=new String[n];

int []grade=new int[n];

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

name[i]=sc.next();

grade[i]=sc.nextInt();

}

String tempname="";

int tempgrade=0;

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

for(int q=0;q<n-1;q++){

if(grade[q]<grade[q+1]){

tempname=name[q];

name[q]=name[q+1];

name[q+1]=tempname;

tempgrade=grade[q];

grade[q]=grade[q+1];

grade[q+1]=tempgrade;

}

}

}

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

for(int j=0;j<15-name[i].length();j++){

System.out.print("");

}

System.out.print(name[i]);

for(int j=0;j<5-String.valueOf(grade[i]).length();j++){

System.out.print("");

}

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

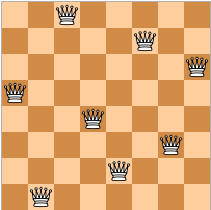
}

}

}

1 八皇后问题2

【问题描述】八皇后问题是一个以国际象棋为背景的问题：如何能够在 8&times;8 的国际象棋棋盘上放置八个皇后，使得任何一个皇后都无法直接吃掉其他的皇后。为了达到此目的，任两个皇后都不能处于同一条横行、纵行或斜线上。例如下图就是八皇后问题的一个解：



假如一棋盘已经正确放置了七个皇后，编写程序求解最后一个皇后的放置位置。

【输入形式】按照棋盘行由小到大的顺序从控制台输入已正确放置的皇后的列数（列数从1开始计数），未放置皇后的行以字符&ldquo;\*&rdquo;表示。各列数和字符&ldquo;\*&rdquo;之间没有任何其它字符分隔，在输入末尾有回车换行符。例如：上图棋盘假如第4行没有放置皇后，其它七个皇后都已放置完毕，则输入的形式为：368\*4752。

【输出形式】在标准输出上输出最后一个皇后应该放置的列数（列数大于等于1，小于等于8，列数后的回车换行可有可无）。若无解，则输出字符串：No Answer。

package p15924;

import java.util.Scanner;

public class P15924 {

public static void main(String[] args) {

Scanner sc=new Scanner(System.in);

int [][]a=new int[8][8];

int i=0,j=0,m=0,n=0,line=0,found=0;

String str=sc.nextLine();

for(i=0;i<8;i++){

for(j=0;j<8;j++){

a[i][j]=0;

}

}

for(i=0;i<8;i++){

char ch=str.charAt(i);

if(ch=='\*'){

line=i;

continue;

}

m=(int)ch-(int)'0'-1;

if(a[i][m]==1){

System.exit(0);

}

for(j=0;j<8;j++){

a[i][j]=1;

a[j][m]=1;

}

for(n=i+1,j=m+1;j<8&&n<8;j++,n++)

a[n][j]=1;

for(n=i-1,j=m-1;j>=0&&n>=0;j--,n--)

a[n][j]=1;

for(n=i+1,j=m-1;j>=0&&n<8;j--,n++)

a[n][j]=1;

for(n=i-1,j=m+1;j<8&&n>=0;j++,n--)

a[n][j]=1;

}

found=0;

for(i=0;i<8;i++)

if(a[line][i]==0)

{

found=1;

System.out.println(i+1);

break;

}

if(found==0)

System.out.println("No Answer");

sc.close();

}

}

2 求小岛面积

【问题描述】用一个二维方阵（最小为3X3，最大为9X9）表示一片海域。方阵中的元素只由0和1组成。1表示海岸线。计算由海岸线围起来的小岛面积（即：由1围起来的区域中0的个数）。如下图所示6X6方阵表示的小岛面积为9：  
0 0 0 1 0 0  
0 0 1 0 1 0  
0 1 0 0 0 1  
1 0 0 0 1 0  
1 0 1 0 1 0  
1 1 0 1 1 1  
上述方阵表示的海域满足下面两个要求：1、小岛只有一个。2、用1表示的海岸线肯定可以封闭成一个小岛，但有可能是凸的，也有可能是凹的。所以在判断时：对于方阵中的任意一个元素0，如果其位于同一行上的两个1之间，并且位于同一列上的两个1之间，则该元素肯定在1围起来的区域中。不符合该规定的其它情况不考虑。

【输入形式】先从标准输入中输入方阵的阶数，然后从下一行开始输入方阵的元素（只会输入0或1），各元素之间以一个空格分隔，每行最后一个元素后没有空格，但会有回车换行符。

【输出形式】在标准输出上输出用整数表示的小岛面积。

package p15950;

import java.util.Scanner;

public class P15950 {

public static void main(String[] args) {

Scanner sc=new Scanner(System.in);

int [][]sea=new int[9][9];

int n=0,i=0,j=0,k=0,found=0,area=0;

n=sc.nextInt();

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

for(j=0;j<n;j++){

sea[i][j]=sc.nextInt();

}

}

area=0;

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

for(j=0;j<n;j++){

if(sea[i][j]==0){

found=0;

for(k=j-1;k>=0;k--){

if(sea[i][k]==1){

found++;

break;

}

}

if(!(found>0)){

continue;

}

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

if(sea[i][k]==1){

found++;

break;

}

}

if(found!=2){

continue;

}

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

if(sea[k][j]==1){

found++;

break;

}

}

if(found!=3){

continue;

}

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

if(sea[k][j]==1){

found++;

break;

}

}

if(found==4){

area++;

}

}

}

}

System.out.println(area);

sc.close();

}

}