ISC LAB MANUAL SOLUTIONS (IMPORTANT PRACTICAL QUESTIONS)

FOR ISC CLASS 11 AND 12

[Year]

CONTENTS

GENERAL PROGRAMS

PRIME TRIPLETS

Q. The three consecutive prime numbers are known as Prime Triplets. if they satisfy the following condition: n, n+2, n+6 are all prime or n, n+4, n+6 are all prime. where n is an integer number>0.
Write a program to input a start limit and end limit and display all the prime triplets in the range. also display the total number of prime triplets.

```
import java.util.Scanner;

class PrimeTriplet
{
    int startLimit, lastLimit;
    PrimeTriplet(int startLimit, int lastLimit)
    {
        this.startLimit = startLimit;
        this.lastLimit = lastLimit;
    }
    public boolean isValid()
    {
        if(startLimit>0 && lastLimit>0)
            return true;
        else
            return false;
    }
    private boolean isPrime(int n)
    {
        for(int i=2;i<n;i++)
        {
            if(n%i==0)
                return false;
        }
        return true;
    }
    public void printAll()
    {
        int count=0;
        System.out.println("Prime Triplets");
        for(int n=startLimit;n<=lastLimit;n++)
        {
            if(isPrime(n))
            {
                if(isPrime(n+6))
                {
                    if(isPrime(n+2))
                    {
                        count++;
                        System.out.println(n +  "\t" +  (n+2) + "\t" + (n+6));
                    }
                    else if(isPrime(n+4))
                    {
                        count++;
                        System.out.println(n +  "\t" +  (n+4) + "\t" + (n+6));
                    }
                }
            }
        }
        System.out.println("Total prime triplet combinations are "+ count );
    }

    public static void main(String args[])
    {
```

```
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the startLimit and last limit: ");
        int startLimit = sc.nextInt();
        int lastLimit = sc.nextInt();
        PrimeTriplet ob = new PrimeTriplet(startLimit, lastLimit);
        if(!ob.isValid())
            System.out.println("Invalid Input");
        else
            ob.printAll();
    }
}
```

UNIQUE-DIGIT INTEGER

Question 3.

A unique digit integer is a positive integer (without leading zeros) with no duplicate digits. For example 7, 135, 214 are all unique digit integers whereas 33, 3121, 300 are not.

Given 2 positive integers m and n. Assume m is less than 30000 and n<30000. You are to output the number of unique-digit integers in the specified range along with their values in the format specified below:

SAMPLE DATA :

INPUT :
m=100
n= 120

OUTPUT :

THE UNIQUE DIGIT INTEGERS ARE :-

102, 103, 104, 105, 106, 107, 108, 109, 120.

FREQUENCY OF UNIQUE DIGIT INTEGERS IS : 9

INPUT :
m=2500
n= 2550

OUTPUT :

THE UNIQUE DIGIT INTEGERS ARE :-

2501, 2503, 2504, 2506, 2507, 2508, 2509, 2510, 2513, 2514, 2516, 2517, 2518,
2519, 2530, 2531, 2534, 2536, 2537, 2538, 2539, 2540, 2541, 2543, 2546,
2547, 2548, 2549.

FREQUENCY OF UNIQUE DIGIT INTEGERS IS : 28

```
import java.util.Scanner;

class UniqueDigitInteger
{
    int m,n;

    public UniqueDigitInteger(int m, int n) {
        this.m = m;
        this.n = n;
    }
    void printAll()
    {
        int freq = 0;
        outerloop: for(int x=m;x<=n;x++)
        {
            for(int div1=x;div1>0;div1=div1/10)
            {
                int digit1 = div1%10;
                int c=0;
                for(int div2=x;div2>0;div2/=10)
                {
                    int digit2 = div2%10;
                    if(digit1 == digit2)
                        c++;
                }
                if(c>1)
                    continue outerloop;
            }
            freq++;
```

```
            System.out.print(freq>1?"," + x:x);
        }
        System.out.println("\nFrequency of unique-digit integers is: " + freq);
    }
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter value for m and n: ");
        int m = sc.nextInt();
        int n = sc.nextInt();
        UniqueDigitInteger ob = new UniqueDigitInteger(m,n);
        ob.printAll();
    }
}
```

CIRCULAR PRIME

A **Circular Prime** is a prime number that remains prime under cyclic shifts of its digits. When the leftmost digit is removed and replaced at the end of the remaining string of digits, the generated number is still prime. The process is repeated until the original number is reached again.
A number is said to be prime if it has only two factors 1 and itself.

Example: 131
311
113

Hence, 131 is a circular prime.

Accept a positive number N and check whether it is a circular prime or not. The new numbers formed after the shifting of the digits should also be displayed.

Test your program with the following data and some random data:

Example 1

INPUT: N = 197

OUTPUT: 197
971
719

197 IS A CIRCULAR PRIME

Example 2

INPUT: N = 1193

OUTPUT: 1193
1931
9311
3119

1193 IS A CIRCULAR PRIME

Example 3

INPUT: N = 29

OUTPUT: 29
92

29 IS NOT A CIRCULAR PRIME

```
import java.util.*;
class CircularPrime
{
    private int n;
    public CircularPrime()
    {
        n = 0;
    }
    public void input()
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the number: ");
        n = sc.nextInt();
```

```
    }
    private boolean isPrime(int n)
    {
        int i;
        for(i=2;i<n;i++)
        {
            if(n%i==0)
                return false;
        }
        return true;
    }
    public void check()
    {
        int x;
        boolean prime=true;
        String s;
        x=n;
        do
        {
            System.out.println(x);
            if(!isPrime(x))
            {
                prime=false;
                break;
            }

            s = ""+x;
            s = s.substring(1) + s.charAt(0);
            x = Integer.parseInt(s);

        }while(x!=n);

        if(prime)
            System.out.println(n + " IS A CIRCULAR PRIME");
        else
            System.out.println(n + " IS NOT A CIRCULAR PRIME");


    }
    public static void main(String args[])
    {
        CircularPrime ob = new CircularPrime();
        ob.input();
        ob.check();
    }
}
```

DIGIT SUM PROBLEM

Given two positive numbers M and N, such that M is between 100 and 10000 and N is less than 100. Find the smallest integer that is greater than M and whose digits add up to N. For example, if M=100 and N=11, then the smallest integer greater than 100 whose digits add up to 11 is 119.

Write a program to accept the numbers M and N from the user and print the smallest required number whose sum of all its digits is equal to N. Also, print the total number of digits present in the required number. The program should check for the validity of the inputs and display an appropriate message for an invalid input.

Test your program with some sample data and some random data:

Example 1:

INPUT : M= 100
N=11

OUTPUT:
The required number is= 119
Total number of digits=3

Example 2

INPUT: M= 1500
N=25

OUTPUT:
The required number is =1699
Total number of digits=4

Example 3

INPUT: M= 99
N=11

OUTPUT:
INVALID INPUT

Example 4

INPUT: M= 112

N=130

OUTPUT:
INVALID INPUT

```
import java.util.Scanner;

class DigitSum
{
    int m, n;
    DigitSum(int m, int n)
    {
        this.m = m;
        this.n = n;
    }
```

```
    boolean isValid()
    {
        if(m>=100 && n<=10000 && n<100)
            return true;
        return false;
    }
    void find()
    {
        int x = m;
        while(true)
        {
            x++;
            int s=0;
            for(int div=x;div>0;div/=10)
            {
                int digit = div%10;
                s+=digit;
            }
            if(s == n)
            {
                System.out.println("The required number = "+ x);
                System.out.println("The total number of digits = " + (x+"").length());
                break;
            }
        }
    }
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the values for M and N: ");
        int m = sc.nextInt();
        int n = sc.nextInt();
        DigitSum ob = new DigitSum(m,n);
        if(!ob.isValid())
            System.out.println("INVALID INPUT");
        else
            ob.find();

    }
}
```

COMPOSITE MAGIC NUMBER

Question 1

A Composite Magic number is a positive integer which is composite as well as a magic number.

Composite number : A composite number is a number which has more than 2 factors.

For example :- 10

Factors are : 1,2,5,10.

Magic number : A magic number is a number in which the eventual sum of the digits is equal to 1.

For example:- 28 = 2+ 8 =10 = 1+0=1

Accept 2 positive integers m and n, where m is less than n as user input. Display the number of composite magic integers that are in the range between m and n (both inclusive) and output them along with the frequency, in the format specified below.

Test your program with the sample data and some random data:

Example 1:

INPUT: m=10

n=100

OUTPUT:

THE COMPOSITE MAGIC INTEGERS ARE :

10,28,46,55,64,82,91,100

FREQUENCY OF COMPOSITE MAGIC INTEGERS IS :8

Example 2:

INPUT: m=1200

n=1300

OUTPUT:

THE COMPOSITE MAGIC INTEGERS ARE :

1207,1216,1225,1234,1243,1252,1261,1270,1288

FREQUENCY OF COMPOSITE MAGIC INTEGERS IS :9

Example 3:

INPUT: m=120

n=99

OUTPUT:

INVALID INPUT

```
import java.util.Scanner;

class CompositeMagic
{
    int m,n;

    public CompositeMagic(int m, int n) {
        this.m = m;
        this.n = n;
    }
    boolean isValid()
    {
        if(m>0 && n>0 && m<n)
            return true;
        return false;
    }
    boolean isComposite(int n)
    {
        for(int i=2;i<n;i++)
            if(n%i==0)
                return true;
        return false;
    }
    boolean isMagic(int n)
    {
        int s=n;
        while(s>9)
        {
            s=0;
            for(int div=n;div>0;div/=10)
            {
                int digit = div%10;
                s+=digit;
            }
```

```
            n=s;
        }
        if(s==1)
            return true;
        return false;
    }
    void generate()
    {
        int freq=0;
        System.out.println("The Composite Magic Integers are: ");
        for(int x=m;x<=n;x++)
        {
            if(isComposite(x) && isMagic(x))
            {
                freq++;
                System.out.print(freq>1?","+x:x);
            }
        }
        System.out.println("\nFrequency of the composite magic integers: " + freq);
    }
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter values for m and n: ");
        int m = sc.nextInt();
        int n = sc.nextInt();
        CompositeMagic ob = new CompositeMagic(m,n);
        if(!ob.isValid())
            System.out.println("INVALID INPUT");
        else
            ob.generate();
    }
}
```

ISBN NUMBER

An ISBN (International Standard Book Number) is a ten digit code which uniquely identifies a book.
The first nine digits represent the Group, Publisher and Title of the book and the last digit is used to check whether ISBN is correct or not.

Each of the first nine digits of the code can take a value between 0 and 9. Sometimes it is necessary to make the last digit equal to ten; this is done by writing the last digit of the code as X.
To verify an ISBN, calculate 10 times the first digit, plus 9 times the second digit, plus 8 times the third and so on until we add 1 time the last digit. If the final number leaves no remainder when divided by 11, the code is a valid ISBN.

For example :

1. 0201103311 = 10*0 + 9 *2 + 8*0 +7 *1 + 6 * 1 + 5 * 0 + 4 * 3 + 3 * 3 + 2 * 1 + 1 * 1 = 55

   Since 55 leaves no remainder when divisible by 11, hence it is a valid ISBN.

2. 007462542X = 10*0 + 9 *0 + 8*7 +7 *4 + 6 *6 + 5 *2 + 4 *5 + 3 * 4 + 2 *2 + 1 * 10 = 176

   Since 176leaves no remainder when divisible by 11, hence it is a valid ISBN.

3. 0112112425 = 10*0 + 9 *1 + 8*1 +7 *2 + 6 * 1 + 5 * 1 + 4 *2 + 3 * 4 + 2 *2 + 1 *5 =71

   Since 55 leaves no remainder when divisible by 11, hence it is a valid ISBN.

Design a program to accept a ten digit code from the user. For an invalid input, display an appropriate message. Verify the code for its validity in the format specified below:

Example 1
INPUT CODE : 0201530821
OUPUT: SUM =99
LEAVES NO REMAINDER – VALID ISBN CODE

Example 2
INPUT CODE: 035680234
OUTPUT: INVALID INPUT

Example 3
INPUT CODE : 0231428031
OUPUT: SUM =122
LEAVES REMAINDER – INVALID ISBN CODE

```
import java.util.*;
class ISBN
{
    String num;
    int sum;
```

```
    public ISBN()
    {
        num="";
        sum=0;
    }
    public void input()
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the ISBN number: ");
        num = sc.next();
    }
    public boolean isValid()
    {
        int l;
        l = num.length();
        return l == 10;
    }
    public void compute()
    {
        int i,m=10,ld;
        for(i=0;i<9;i++)
        {
            sum += (num.charAt(i)-48)*m;
            m--;
        }

        ld = num.charAt(9);
        if(ld == 'X')
            ld = 10;
        else
            ld = ld - 48;

        sum += ld;
    }
    public void display()
    {
        System.out.println("SUM=" + sum);
        if(sum%11 == 0)
            System.out.println("LEAVES NO REMAINDER - VALID ISBN CODE");
        else
            System.out.println("LEAVES REMAINDER - INVALID ISBN CODE");
    }
    public static void main(String args[])
    {
        ISBN ob = new ISBN();
        ob.input();
        if(ob.isValid())
        {
            ob.compute();
            ob.display();
        }
        else
        {
            System.out.println("INVALID INPUT");
        }
    }
}
```

PRIME PALINDROME INTEGER

Question 1.

A Prime – palindrome integer is a positive integer (without leading zeros) which is prime as well as palindrome. Given 2 positive integers m and n, where m<n, write a program to determine how many prime palindrome integers are there in the range between m and n (both inclusive) and output them.

The input contains two positive integers m and n, where m < 3000 and n< 3000. Display the number of prime palindrome integers in the specified range along with their values in the format specified below:

Test your program with the sample data and some random data:

Example 1

INPUT:
m = 100
n= 1000

OUTPUT:
THE PRIME PALINDROME INTEGERS ARE:
101, 131, 151, 181, 191, 313, 353, 373, 383, 727, 757, 787, 797, 919, 929
FREQUENCY OF PRIME PALINDROME INTEGERS : 15

Example 2

INPUT:
m = 100
n= 1000

OUTPUT:
OUT OF RANGE

```
import java.util.*;
class PrimePalindrome
{

    private int m,n;
    public PrimePalindrome()
    {
        m=n=0;
    }
    public void input()
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the range (m,n): ");
        m = sc.nextInt();
        n = sc.nextInt();
    }
    public boolean isValid()
    {
        if(m<300 && n<3000)
            return true;
        else
            return false;
    }
    public void display()
    {
        int x,i,div,digit, rev,nf,npp=0;

        System.out.println("THE PRIME PALINDROME INTEGERS ARE: ");
        for(x=m;x<=n;x++)
        {
            nf=0;
            for(i=1;i<=x;i++)
```

```
            {
                if(x%i==0)
                    nf++;
            }

            rev=0;
            for(div=x;div>0;div/=10)
            {
                digit =div%10;
                rev = rev*10+digit;
            }

            if(nf==2 && rev == x)
            {
                if(npp>0)
                    System.out.print(",");
                System.out.print(x);
                npp++;
            }
        }
        System.out.print("\nFREQUENCY OF PRIME PALINDROME INTEGERS: "+ npp);
    }

    public static void main(String args[])
    {
        PrimePalindrome ob = new PrimePalindrome();
        ob.input();
        if(ob.isValid())
            ob.display();
        else
            System.out.println("OUT OF RANGE");
    }
}
```

DISPLAY NUMBER IN WORDS

Question 1.

Write a program to input a natural number less than 1000 and output it in words. Test your program for the following set of data:

INPUT	29
OUTPUT	TWENTY NINE
INPUT	17001
OUTPUT	OUT OF RANGE
INPUT	119
OUTPUT	ONE HUNDRED AND NINETEEN
INPUT	500
OUTPUT	FIVE HUNDRED

```
import java.util.*;
class NumberToWords
{
    int n;
    NumberToWords(int n)
    {
        this.n = n;
    }
    boolean isValid()
    {
        if(n>0 && n<1000)
            return true;
        return false;
    }
    void convert()
    {
        String
names1[]={"ZERO","ONE","TWO","THREE","FOUR","FIVE","SIX","SEVEN","EIGHT","NINE"};
        String
names2[]={"TEN","ELEVEN","TWELVE","THIRTEEN","FOURTEEN","FIFTEEN","SIXTEEN","SEVENTEEN"
,"EIGHTEEN","NINETEEN"};
        String
names3[]={"TWENTY","THIRTY","FOURTY","FIFTY","SIXTY","SEVENTY","EIGHTY","NINETY"};

        int nd = (n+"").length();
        int arr[]=new int[nd];
        int div,i=0;
        for(div = n;div>0;div=div/10)
            arr[i++]= div%10;

        int d;
        for(i= nd-1;i>=0;i--)
        {
            d = arr[i];

            if(i==0)
            {
                if(d>=0&&d<=9)
                    System.out.print (names1[d]+ " ");
            }
            else if(i==1)
            {
                if(d==1)
                {
                    d=arr[--i];
                    System.out.print(names2[d]+ " ");
                }
```

```
                else
                {
                    System.out.print(names3[d-2]+ " ");
                    while(i>0&&arr[i-1]==0)
                        i--;
                }
            }
            else if(i==2)
            {
                System.out.print(names1[d]+ " HUNDRED ");
                while(i>0&&arr[i-1]==0)
                        i--;
                if(i>0)
                        System.out.print("AND ");
            }

        }

    }
    public static void main(String Args[])
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the number: ");
        int n = sc.nextInt();
        NumberToWords ob = new NumberToWords(n);
        if(!ob.isValid())
            System.out.println("OUT OF RANGE");
        else
            ob.convert();

    }
}
```

VALID DATE

Question 3.

Design a program which accepts your date of birth in dd mm yyyy format. Check whether the date entered is a valid date or not. If it is valid, display "VALID DATE". Also compute and display the day number of the year for the date of birth. If it is invalid display "INVALID DATE" and then terminate the program.

Test your program with sample data and some random data:

Example 1.

INPUT : Enter your date of birth in dd mm yyyy format
05
01
2010

OUTPUT: VALID DATE
5

Example 2.

INPUT : Enter your date of birth in dd mm yyyy format
03
04
2010

OUTPUT: VALID DATE
93

Example 3.

INPUT : Enter your date of birth in dd mm yyyy format
34
06
2010

OUTPUT: INVALID DATE

```
import java.util.Scanner;

class DateValidity
{
    int dd, mm, yyyy;
    DateValidity(int dd, int mm, int yyyy)
    {
        this.dd = dd;
        this.mm = mm;
        this.yyyy = yyyy;
    }
    boolean isLeapYear(int year)
    {
        if(year%100==0)
        {
            if(year%400 == 0)
                return true;
        }
        else if(year%4 == 0)
            return true;
        return false;
    }
    boolean isValid()
    {
        if(yyyy<0)
            return false;
        if(mm <1 || mm>12)
            return false;
```

```
        int days[]={31,28,31,30,31,30,31,31,30,31,30,31};
        if(isLeapYear(yyyy))
            days[1]=29;
        if(dd>days[mm-1])
            return false;
        return true;
    }
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter your date of birth in dd mm yyyy format");
        int dd = sc.nextInt();
        int mm = sc.nextInt();
        int yyyy = sc.nextInt();
        DateValidity ob = new DateValidity(dd,mm,yyyy);
        if(!ob.isValid())
            System.out.println("INVALID DATE");
        else
            System.out.println("VALID DATE\n"+dd);

    }
}
```

DATE FORMAT

Input project allotment date and number of days after which the project must be submitted, find the submission date of the project.

```
import java.util.Scanner;
class AddDaysToDate
{
    int dd, mm, yyyy;
    int nd;

    public AddDaysToDate(int dd, int mm, int yyyy, int nd) {
        this.dd = dd;
        this.mm = mm;
        this.yyyy = yyyy;
        this.nd = nd;
    }
    boolean isLeapYear(int year)
    {
        if(year%100==0)
        {
            if(year%400 == 0)
                return true;
        }
        else if(year%4 == 0)
            return true;
        return false;
    }
    void add()
    {
        int days[]={31,28,31,30,31,30,31,31,30,31,30,31};
        int daysLeft = nd;
        while(daysLeft > 0)
        {
            if(isLeapYear(yyyy))
                days[1]=29;
            while(mm < 13 && (daysLeft-(days[mm-1]-dd))>0)
            {
                daysLeft-=(days[mm-1]-dd);
                dd=0;
                mm++;
            }
            if(mm==13)
            {
                mm=1;
                yyyy++;
            }
            else
               break;
        }
        dd=daysLeft;

    }
    void display()
    {
        System.out.println("Project Submission Date: ");
        System.out.println(dd + "/" + mm + "/" + yyyy);
    }

    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the Project Allotment Date: ");
        int dd = sc.nextInt();
        int mm = sc.nextInt();
```

```
        int yyyy = sc.nextInt();
        System.out.println("Enter the no of days to add: ");
        int nd = sc.nextInt();
        AddDaysToDate ob = new AddDaysToDate(dd,mm,yyyy,nd);
        ob.add();
        ob.display();
    }
}
```

STRING PROGRAMS

WORDS START AND END WITH VOWEL

Write a program to accept a sentence which may be terminated by either '.', '?' or '!' only. The words may be separated by more than one blank space and are in UPPER CASE.

Perform the following tasks:

(a) Find the number of words beginning and ending with a vowel.

(b) Place the words which begin and end with a vowel at the beginning, followed by the remaining words as they occur in the sentence.

Test your program with the sample data and some random data:

Example 1

INPUT: ANAMIKA AND SUSAN ARE NEVER GOING TO QUARREL ANYMORE.

OUTPUT: NUMBER OF WORDS BEGINNING AND ENDING WITH A VOWEL = 3

ANAMIKA ARE ANYMORE AND SUSAN NEVER GOING TO QUARREL

Example 2

INPUT: YOU MUST AIM TO BE A BETTER PERSON TOMORROW THAN YOU ARE TODAY.

OUTPUT: NUMBER OF WORDS BEGINNING AND ENDING WITH A VOWEL = 2

A ARE YOU MUST AIM TO BE BETTER PERSON TOMORROW THAN YOU TODAY

Example 3

INPUT: LOOK BEFORE YOU LEAP.

OUTPUT: NUMBER OF WORDS BEGINNING AND ENDING WITH A VOWEL = 0

LOOK BEFORE YOU LEAP.

Example 4

INPUT: HOW ARE YOU@

OUTPUT: INVALID INPUT

```
import java.util.Scanner;

class StartEndVowel
{
    String s;
    StartEndVowel(String s)
    {
        this.s = s;
    }
    boolean isValid()
    {
        char ch = s.charAt(s.length()-1);
        if(ch=='.'||ch=='?'||ch=='!')
            return true;
        return false;
    }
    boolean isVowel(char ch)
    {
        String vowels="AEIOU";
        for(int i=0;i<vowels.length();i++)
            if(ch == vowels.charAt(i))
                return true;
```

```
        return false;
    }
    void process()
    {
        int count=0;
        String vs = "", cs = "";
        char terminator = s.charAt(s.length()-1);
        s = s.substring(0,s.length()-1);
        Scanner sc = new Scanner(s);
        while(sc.hasNext())
        {
            String word = sc.next();
            if(isVowel(word.charAt(0)) && isVowel(word.charAt(word.length()-1)))
            {
                vs += word + " ";
                count++;
            }
            else
            {
                cs += word + " ";
            }
        }
        System.out.println("NUMBER OF WORDS BEGINNING AND ENDING WITH VOWEL = "+
count);
        System.out.println(vs + cs.substring(0,cs.length()-1) + terminator);

    }
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the sentence: ");
        String s = sc.nextLine();
        StartEndVowel ob = new StartEndVowel(s);
        if(!ob.isValid())
            System.out.println("INVALID INPUT");
        else
            ob.process();
    }
}
```

TITLE CASE, VOWELS, CONSONANTS COUNT

Write a program to accept a sentence which may be terminated by either '.' Or '?' only. The words are to be separated by a single blank space. Print an error message if the input does not terminate with '.' Or '?'. You can assume that no word in the sentence exceeds 15 characters, so that you get a proper formatted output. Perform the following tasks:

i) Convert the first letter of each word to uppercase.
ii) Find the number of vowels and consonants in each word and display them with proper heading along with the words.

Test your program with the following inputs:

Example 1

INPUT: Intelligence plus character is education.

OUTPUT:

Intelligence plus character is education.

Word	Vowels	Consonants
Intelligence	5	7
Plus	1	3
Character	3	6
Is	1	1
Education	5	4

Example 2

INPUT: God is great.

OUTPUT:

God is great

Word	Vowels	Consonants
God	1	2
Is	1	1
Great	2	3

```
import java.util.Scanner;

class TitleCase
{
    String s;
    TitleCase(String s)
    {
        this.s = s;
    }
    boolean isValid()
    {
        char ch = s.charAt(s.length()-1);
        if(ch=='.'||ch=='?')
            return true;
        return false;
    }
    boolean isVowel(char ch)
    {
        String vowels="aeiou";
        ch = Character.toLowerCase(ch);
```

```
        for(int i=0;i<vowels.length();i++)
            if(ch == vowels.charAt(i))
                return true;
        return false;
    }
    void process()
    {
        char terminator = s.charAt(s.length()-1);
        s = s.substring(0,s.length()-1);
        Scanner sc;
        String result = "";
        sc = new Scanner(s);
        while(sc.hasNext())
        {
            String word = sc.next();
            word = Character.toUpperCase(word.charAt(0)) + word.substring(1);
            result += word + " ";
        }
        System.out.println(result.substring(0,result.length()-1)+terminator);
        System.out.printf("\n%-15s%-15s%-15s\n","Word","Vowels","Consonants");
        sc = new Scanner(s);
        while(sc.hasNext())
        {
            String word = sc.next();
            int v=0,c=0;
            for(int i=0;i<word.length();i++)
            {
                if(isVowel(word.charAt(i)))
                    v++;
                else
                    c++;
            }
            System.out.printf("%-15s%-15d%-15d\n",word,v,c);
        }

    }
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the sentence: ");
        String s = sc.nextLine();
        TitleCase ob = new TitleCase(s);
        if(!ob.isValid())
            System.out.println("INVALID INPUT");
        else
            ob.process();
    }
}
```

REDUCE SPACE, DELETE WORD

Write a program to accept a sentence which may be terminated by either '.', '?' or '!' only. Any other character may be ignored. The words may be separated by more than one blank space and are in UPPER CASE.

Perform the following tasks:

a) Accept a sentence anduce all the extra blank space between 2 words to a single blank space.

b) Accept a word from the user which is a part of the sentence along with its position number and delete the word and display the sentence.

Test your program for the following data and some random data.:

Example 1

INPUT: A MORNING WALK IS A BLESSING FOR THE WHOLE DAY.

WORD TO BE DELETED : IS

WORD POSITION IN THE SENTENCE : 6

OUTPUT: A MORNING WALK IS A BLESSING FOR THE WHOLE DAY.

Example 2

INPUT: AS YOU SOW, SO SO YOU REAP.

WORD TO BE DELETED : SO

WORD POSITION IN THE SENTENCE : 4

OUTPUT: AS YOU SOW, SO YOU REAP.

Example 3

INPUT: STUDY WELL ##.

OUTPUT: INVALID INPUT.

```
import java.util.Scanner;

class DeleteWord
{
    String s,w;
    int p;
    DeleteWord(String s, String w , int p)
    {
        this.s = s;
        this.w = w;
        this.p = p;
    }
    boolean isValid()
    {
        char ch = s.charAt(s.length()-1);
        if(ch=='.'||ch=='?'||ch=='!')
            return true;
        return false;
    }
    void process()
    {
        int count=0;
        String result="";
        char terminator = s.charAt(s.length()-1);
        s = s.substring(0,s.length()-1);
        Scanner sc = new Scanner(s);
        int i=0;
        while(sc.hasNext())
        {
            i++;
            String word = sc.next();
            if(!(i==p && w.equals(word)))
                result = result + word + " ";
        }
        System.out.println(result.substring(0,result.length()-1)+ terminator);

    }
    public static void main(String args[])
    {
```

```
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the sentence: ");
        String s = sc.nextLine();
        System.out.println("Word to be deleted: ");
        String w = sc.nextLine();
        System.out.println("Enter the position in the sentence: ");
        int p = Integer.parseInt(sc.nextLine());
        DeleteWord ob = new DeleteWord(s,w,p);
        if(!ob.isValid())
            System.out.println("INVALID INPUT");
        else
            ob.process();
    }
}
```

COUNT PALINDROME WORDS

A palindrome is a word that may be read the same way in either direction. Accept a sentence in UPPERCASE which is terminated by either ".", "?", or "!". Each word of the sentence is separated by a single blank space.

Perform the following tasks:

a) Display the count of palindromic words in the sentence.

b) Display the palindromic words in the sentence

Example of palindromic words:

MADAM, ARORA, NOON

Test your program with the sample data and some random data:

Example 1

INPUT : MOM AND DAD ARE COMING AT NOON.

OUTPUT : MOM DAD NOON
NUMBER OF PALINDROMIC WORDS : 3

Example 2

INPUT : NITIN ARORA USES LIRIL SOAP.

OUTPUT : NITIN ARORA LIRIL
NUMBER OF PALINDROMIC WORDS : 3

Example 3

INPUT : HOW ARE YOU?

OUTPUT : NO PALINDROMIC WORDS

```
import java.util.Scanner;

class CountPalindrome
{
    String s;
    CountPalindrome(String s)
    {
        this.s = s;
    }
    boolean isValid()
    {
        char ch = s.charAt(s.length()-1);
        if(ch=='.'||ch=='?'||ch=='!')
            return true;
        return false;
    }
```

```
    boolean isPalindrome(String s)
    {
        int l = s.length();
        int m=l/2;
        for(int i=0;i<m;i++)
            if(s.charAt(i)!=s.charAt(l-1-i))
                return false;
        return true;
    }
    void process()
    {
        int count=0;
        s = s.substring(0,s.length()-1);
        Scanner sc = new Scanner(s);
        while(sc.hasNext())
        {
            String word = sc.next();
            if(isPalindrome(word))
            {
                System.out.print(word + " ");
                count++;
            }
        }
        System.out.println("\nNumber of Palindromic words: " + count);

    }
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the sentence: ");
        String s = sc.nextLine();
        CountPalindrome ob = new CountPalindrome(s);
        if(!ob.isValid())
            System.out.println("INVALID INPUT");
        else
            ob.process();
    }
}
```

ARRANGE SENTENCE IN ALPHABETIC ORDER

Question 2.

Write a program to accept a sentence as input. The words in the string are to be separated by a blank. Each word must be in uppercase. The sentence is terminated by either ".", "!", or "?". Perform the following tasks:

i) Obtain the length of the sentence (measured in words).
ii) Arrange the sentence in alphabetical order of words.

Test your program with some sample data and some random data:

Example 1

INPUT:
NECESSITY IS THE MOTHER OF INVENTION.

OUTPUT:
LENGTH : 6
REARRANGED SENTENCE

INVENTION IS MOTHER NECESSITY OF THE

Example 2

INPUT:
BE GOOD TO OTHERS.
OUTPUT:
LENGTH : 4
REARRANGED SENTENCE

BE GOOD OTHERS TO

```
import java.util.Scanner;

class AlphabeticOrder
{
    String s;
    AlphabeticOrder(String s)
    {
        this.s = s;
    }
    boolean isValid()
    {
        char ch = s.charAt(s.length()-1);
        if(ch=='.'||ch=='?'||ch=='!')
            return true;
        return false;
    }
    boolean isPalindrome(String s)
    {
        int l = s.length();
        int m=l/2;
        for(int i=0;i<m;i++)
            if(s.charAt(i)!=s.charAt(l-1-i))
                return false;
        return true;
    }
    void process()
    {
        int count=0;
```

```
        s = s.substring(0,s.length()-1);
        Scanner sc = new Scanner(s);
        while(sc.hasNext())
        {
            String word = sc.next();
            count++;
        }

        String words[] = new String[count];
        sc = new Scanner(s);
        int i=0;
        while(sc.hasNext())
        {
            words[i++]=sc.next();
        }

        for(i=1;i<words.length;i++)
        {
            for(int j=0;j<words.length-i;j++)
            {
                if(words[j].compareTo(words[j+1])>0)
                {
                    String t = words[j];
                    words[j] = words[j+1];
                    words[j+1] = t;
                }
            }
        }

        System.out.println("Length: " + words.length);
        System.out.println("Rearranged Sentence: ");
        for(i=0;i<words.length;i++)
            System.out.print(words[i] + " ");
    }
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the sentence: ");
        String s = sc.nextLine();
        AlphabeticOrder ob = new AlphabeticOrder(s);
        if(!ob.isValid())
            System.out.println("INVALID INPUT");
        else
            ob.process();
    }
}
```

WORD FEQUENCY

Question 3.

Input a paragraph containing 'n' number of sentences where (1=<n<4). The words are to be separated with a single blank space and are in upper-case. A sentence may be terminated either with a full stop '.' or a question mark'?' only. Any other character may be ignored. Perform the following operations:

i) Accept the number of sentences. If the number of sentences exceeds the limit, an appropriate error message must be displayed.
ii) Find the number of words in the whole paragraph.
iii) Display the words in ascending order of their frequency. Words with same frequency may appear in any order.

Example 1.

INPUT :

```
Enter number of sentences.
1
Enter sentences.
TO BE OR NOT TO BE.
```

OUTPUT

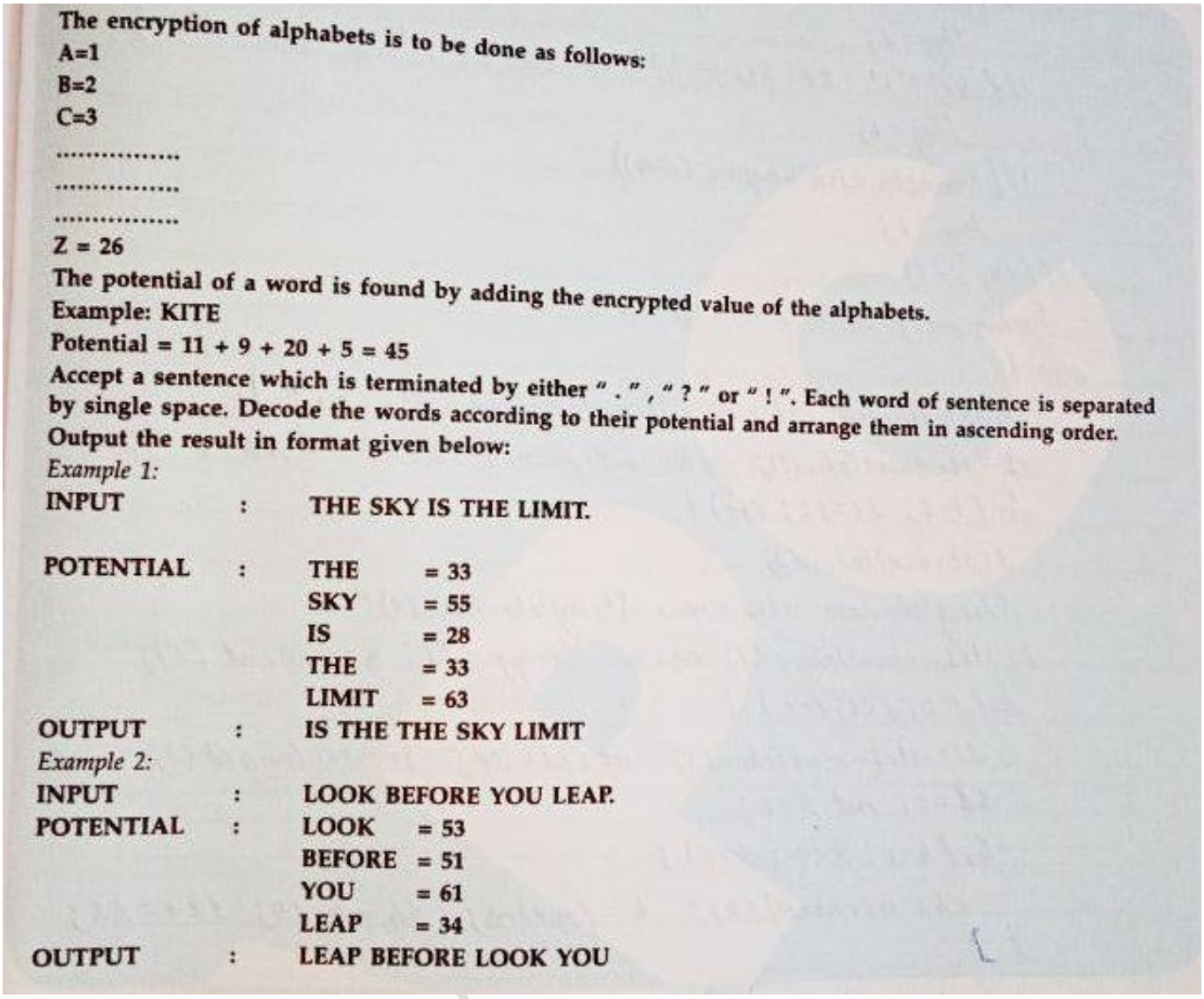
```
Total number of words  :    6

WORD                    FREQUENCY
OR                          1
NOT                         1
TO                          2
BE                          2
```

```java
import java.util.Scanner;

class WordFrequency
{
    int n;
    String s;
    WordFrequency(int n)
    {
        this.n = n;
    }
    boolean isValid()
    {
        if(n>=1 && n<=4)
            return true;
        else
            return false;
    }
    void input()
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter sentences: ");
        s = sc.nextLine();
    }
    void process()
    {
        int count=0;
        Scanner sc = new Scanner(s);
        while(sc.hasNext())
        {
            String word = sc.next();
            int wl = word.length();
            char lc = word.charAt(wl-1);
```

```
            if(lc == '?' || lc == '.')
                word = word.substring(0,wl-1);
            count++;
        }
        System.out.println("Total number of words: " + count);
        String words[] = new String[count];
        int freq[] = new int[count];
        sc = new Scanner(s);
        int count2=0;
        while(sc.hasNext())
        {
            String word = sc.next();
            int wl = word.length();
            char lc = word.charAt(wl-1);
            if(lc == '?' || lc == '.'||lc == ',')
                word = word.substring(0,wl-1);
            int j;
            for(j=0;j<count2;j++)
            {
                if(words[j].equals(word))
                {
                    freq[j]++;
                    break;
                }
            }
            if(j == count2)
            {
                words[count2]=word;
                freq[count2++] = 1;
            }
        }

        for(int i=1;i<count2;i++)
        {
            for(int j=0;j<count2-i;j++)
            {
                if(freq[j]>freq[j+1])
                {
                    String ts = words[j];
                    words[j]= words[j+1];
                    words[j+1]=ts;

                    int ti = freq[j];
                    freq[j] = freq[j+1];
                    freq[j+1] = ti;
                }
            }
        }
        System.out.printf("%-15s%-15s\n","Word","Frequency");
        for(int j=0;j<count2;j++)
        {
            System.out.printf("%-15s%-15d\n",words[j],freq[j]);
        }

    }
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter number of sentences: ");
        int n = Integer.parseInt(sc.nextLine());
        WordFrequency ob = new WordFrequency(n);
        if(!ob.isValid())
            System.out.println("INVALID INPUT");
        else
        {
```

```
            ob.input();
            ob.process();
        }
    }
}
```

ENCRYPTION



The encryption of alphabets is to be done as follows:
A=1
B=2
C=3
.....
.....
.....
Z = 26
The potential of a word is found by adding the encrypted value of the alphabets.
Example: KITE
Potential = 11 + 9 + 20 + 5 = 45
Accept a sentence which is terminated by either " . " , " ? " or " ! ". Each word of sentence is separated by single space. Decode the words according to their potential and arrange them in ascending order.
Output the result in format given below:
Example 1:

INPUT	:	THE SKY IS THE LIMIT.	
POTENTIAL	:	THE	= 33
		SKY	= 55
		IS	= 28
		THE	= 33
		LIMIT	= 63
OUTPUT	:	IS THE THE SKY LIMIT	

Example 2:

INPUT	:	LOOK BEFORE YOU LEAP.	
POTENTIAL	:	LOOK	= 53
		BEFORE	= 51
		YOU	= 61
		LEAP	= 34
OUTPUT	:	LEAP BEFORE LOOK YOU	

```
import java.util.Scanner;
import java.util.StringTokenizer;

class Encryption
{
    String s;
    Encryption(String s)
    {
        this.s = s;
    }
    boolean isValid()
    {
        char ch = s.charAt(s.length()-1);
        if(ch=='.'||ch=='?'||ch=='!')
            return true;
        return false;
    }
    void process()
    {
        s = s.substring(0,s.length()-1);
        StringTokenizer st = new StringTokenizer(s);
        int wordCount = st.countTokens();
        String words[] = new String[wordCount];
```

```
        int potentials[] = new int[wordCount];
        int n = 0;
        while(st.hasMoreTokens())
        {
            String word = st.nextToken();
            int potential = 0;
            for(int i=0;i<word.length();i++)
            {
                potential += (int)(word.charAt(i)-64);
            }
            System.out.println(word + " = " + potential);
            words[n]= word;
            potentials[n++] = potential;
            int i;
            for(i=n-1;i>0 && potentials[i-1]>potentials[i];i--)
            {
                int p = potentials[i];
                potentials[i]=potentials[i-1];
                potentials[i-1]=p;

                String w = words[i];
                words[i] = words[i-1];
                words[i-1] = w;
            }

        }

        for(int i = 0;i<wordCount;i++)
        {
            System.out.print(words[i] + " ");
        }

    }
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the sentence: ");
        String s = sc.nextLine();
        Encryption ob = new Encryption(s);
        if(!ob.isValid())
            System.out.println("INVALID INPUT");
        else
            ob.process();
    }
}
```

ANAGRAMS

Question 3.

We would like to generate all possible anagrams of a word. For example if the given word is "TOP", there will be 6 possible anagrams:
TOP, TPO, OPT, OTP, PTO, POT

An anagram must be printed only once. You may output the anagram in any order. Also output the total number of anagrams. You may assume that the number of letters, N, in the word will be 7 at most, i.e. N<=7

Test your program for the given data and some random data.

Sample Date:
Input:
TO

Output:
TO, OT
Total number of anagrams = 2

Input:
LEAN

Output:
LEAN, LENA, LAEN, LANE, LNEA, LNAE, EALN, EANL, ELAN, ELNA, ENLA, ENAL, ALNE, ALEN
ANLE, ANEL, AENL, AELN, NLEA, NLAE, NELA, NEAL, NALE, NAEL

Total number of anagrams = 24

```java
import java.util.Scanner;

class Anagram
{
    String w;
    int count;
    Anagram(String w)
    {
        this.w = w;
        count = 0;
    }
    void print()
    {
        System.out.println("Anagrams of the word " + w + " are: ");
        anagram(w.getBytes(),0,w.length()-1);
        System.out.println("Total Number Of Anagrams: " + count);
    }
    void swap(byte arr[], int i, int j)
    {
        byte t = arr[i];
        arr[i] = arr[j];
        arr[j] = t;
    }
    void anagram(byte arr[], int l, int r)
    {
        if (l == r)
        {
```

```
            count++;
            System.out.println(new String(arr));
        }
        else
        {
            for (int i = l; i <= r; i++)
            {
                swap(arr,l,i);
                anagram(arr, l+1, r);
                swap(arr,l,i);
            }
        }
    }
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the word: ");
        String w = sc.next();
        Anagram ob = new Anagram(w);
        ob.print();
    }
}
```

SHIFTING ENCRYPTION

A simple encryption system uses a shifting process to hide a message . The value of the shift can be in the range of 1 to 26. For example a shift of 7 means that A=U, B=V, C=W, etc. i.e.

Text : A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Code : U V W X Y Z A B C D E F G H I J K L M N O P Q R S T

First an extra space is added to the end of the string. To make things a little more difficult, spaces within the original text are replaced with QQ before the text is encrypted. Double Q(QQ) was selected because no English word ends in Q or contains QQ.

Additionally the coded message is printed in blocks of six characters separated by spaces. The last block might not contain six characters. Write a program that takes the coded text (less than 100 characters), the shift value and prints the decoded original text. Your program must reject any non valid value for shift and display an error message " INVALID SHIFT VALUE". Assume all characters are upper-case. Test your program for the following data and some data that you have coded, using the rules given above:

SAMPLE DATA:

INPUT:
CODED TEXT: UHINBY LKKQCH HYLKK
SHIFT : 7
OUTPUT
DECODED TEXT : ANOTHER WINNER

INPUT:
CODED TEXT: RUIJGG EVGGBK SAGG
SHIFT : 11
OUTPUT
DECODED TEXT : BEST OF LUCK

INPUT:
CODED TEXT: DKSMMW NAMMUK QMM
SHIFT : 29
OUTPUT
INVALID SHIFT VALUE

```
import java.util.Scanner;
import java.util.StringTokenizer;

class ShiftingEncryption
{
    String text;
    int shift;
    ShiftingEncryption(String text, int shift)
    {
        this.text = text;
        this.shift = shift;
    }
    boolean isValid()
    {
        if(shift >= 1 && shift <=26)
            return true;
        else
            return false;
    }
    char unshift(char ch)
    {
        ch += (shift-1);
        if(ch > 90)
            ch -= 26;
        return ch;
    }
    void process()
```

```
    {
        String message = text.replace(" ", "");
        int l=message.length();
        String pass1="";
        for(int i=0;i<l;i++)
        {
            pass1 += unshift(message.charAt(i));
        }
        String pass2 = pass1.replace("QQ"," ");
        System.out.println("DECODED TEXT: " + pass2);
    }
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the coded text: ");
        String text = sc.nextLine();
        System.out.println("Enter the shift value: ");
        int shift = Integer.parseInt(sc.nextLine());
        ShiftingEncryption ob = new ShiftingEncryption(text,shift);
        if(!ob.isValid())
            System.out.println("INVALID SHIFT VALUE");
        else
            ob.process();
    }
}
```

CONVERT TIME TO WORDS

Question 2.

Given a time in numbers we can convert it into words. For example:-

5:00	five o'clock
5:10	ten minutes past five
5:15	quarter past five
5:30	half past five
5:40	twenty minutes to six
5:45	quarter to six
5:47	thirteen minutes to six

Write a program which first inputs two integers, the first between 1 and 12 (both inclusive) and second between 0 and 59(both inclusive) and then prints out the time they represent, in words. Your program should follow the format of the examples below:-

SAMPLE DATA :

INPUT:
TIME : 3, 0
OUTPUT: 3:00 three o'clock

INPUT:
TIME : 7,29
OUTPUT: 7:29 twenty minutes past seven

INPUT:
TIME : 6,34
OUTPUT: 6:34 twenty minutes to seven

INPUT:
TIME : 12,1
OUTPUT: 12:01 one minute past twelve

INPUT:
TIME : 12,45
OUTPUT: 12:45 quarter to 9

INPUT:
TIME : 10,59
OUTPUT: 10:59 one minute to eleven

INPUT:
TIME : 14, 60
OUTPUT: incorrect input

Test your program for the data values given in the examples above and some random data.

```
import java.util.Scanner;

class TimeToWords
{
    int hour, minute;
    TimeToWords(int hour, int minute)
    {
        this.hour = hour;
        this.minute = minute;
    }
    boolean isValid()
    {
        if(hour>=1 && hour<=12 && minute>=0 && minute<=59)
            return true;
        return false;
    }
    String getName(int n)
    {
```

```
        String
names[]={"zero","one","two","three","four","five","six","seven","eight","nine","ten",

"eleven","twelve","thirteen","fourteen","fifteen","sixteen","seventeen","eighteen","nin
eteen","twenty"};
        if(n<=20)
            return names[n];
        else
            return names[20] + " " + names[n%10];
    }
    String titleCase(String s)
    {
        return Character.toUpperCase(s.charAt(0)) + s.substring(1);
    }
    void convert()
    {
        String result;
        if(minute == 0)
            result = getName(hour) + " o' clock";
        else
        {
            String toOrPast;
            if(minute<=(60-minute))
                toOrPast = "past ";
            else
            {
                minute = 60-minute;
                hour++;
                toOrPast = "to ";
            }
            if(minute == 15)
                result = "quarter "+toOrPast + getName(hour);
            else if(minute == 30)
                result = "half "+toOrPast + getName(hour);
            else if(minute == 1)
                result = "one minute " + toOrPast + getName(hour);
            else
                result = getName(minute)+" minutes " + toOrPast + getName(hour);
        }
        System.out.println(titleCase(result));
    }
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the hour: ");
        int hour = sc.nextInt();
        System.out.println("Enter the minute: ");
        int minute = sc.nextInt();
        TimeToWords ob = new TimeToWords(hour,minute);
        if(!ob.isValid())
            System.out.println("Incorrect Input");
        else
            ob.convert();
    }
}
```

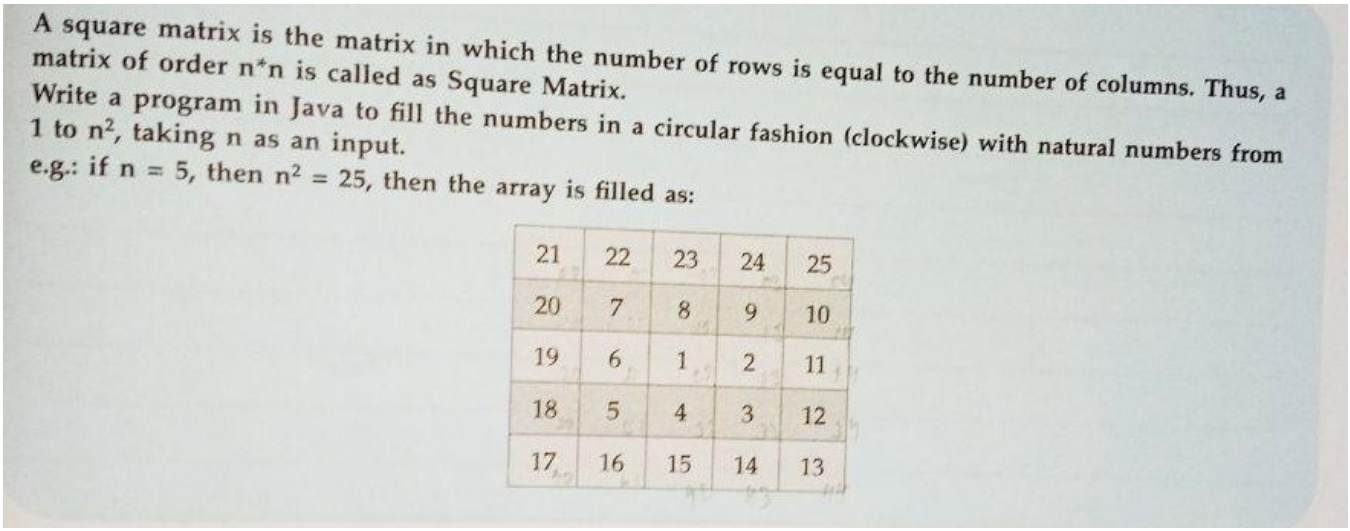
ARRAYS

PRODUCT OF MATRICES

Write a program to accept integers in two matrices A and B or order 3x3 and find their product.

```
import java.util.Scanner;

class MatrixProduct
{
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        int a[][] = new int[3][3];
        int b[][] = new int[3][3];
        int c[][] = new int[3][3];

        int i,j,k;
        System.out.println("Enter elements for first matrix: ");
        for(i=0;i<3;i++)
            for(j=0;j<3;j++)
                a[i][j] = sc.nextInt();
        System.out.println("Enter elements for second matrix: ");
        for(i=0;i<3;i++)
            for(j=0;j<3;j++)
                b[i][j] = sc.nextInt();
        int s;
        for(i=0;i<3;i++)
            for(j=0;j<3;j++)
            {
                s=0;
                for(k=0;k<3;k++)
                    s+=a[i][k] * b[k][j];
                c[i][j]=s;
            }

        System.out.println("The product of two matrices is: ");
        for(i=0;i<3;i++)
        {
            for(j=0;j<3;j++)
                System.out.print(c[i][j]+ "\t");
            System.out.println();
        }
    }
}
```

SPIRAL MATRIX



A square matrix is the matrix in which the number of rows is equal to the number of columns. Thus, a matrix of order n*n is called as Square Matrix.
Write a program in Java to fill the numbers in a circular fashion (clockwise) with natural numbers from 1 to n^2, taking n as an input.
e.g.: if n = 5, then n^2 = 25, then the array is filled as:

21	22	23	24	25
20	7	8	9	10
19	6	1	2	11
18	5	4	3	12
17	16	15	14	13

```
class SpiralMatrix
{
    public static void main(String args[])
    {
        int n=5;
        int m[][] = new int[n][n];
        int i,j,k=1;
        if(n%2==1)
            m[n/2][n/2]=k++;
        for(i=n/2;i>=0;i--)
        {
            for(j=i+1;j<=n-1-i;j++)
                m[j][n-1-i]=k++;
            for(j=n-2-i;j>=i;j--)
                m[n-1-i][j]=k++;
            for(j=n-2-i;j>=i;j--)
                m[j][i]=k++;
            for(j=i+1;j<=n-1-i;j++)
                m[i][j]=k++;
        }
        System.out.println("The matrix is: ");
        for(i=0;i<n;i++)
        {
            for(j=0;j<n;j++)
            {
                System.out.print(m[i][j] + "\t");
            }
            System.out.println();
        }
    }
}
```

SORT NON-BOUNDARY ELEMENTS

Write a program to declare a square matrix A[ ] [ ] of order (M×M) where 'M' must be greater than 3 and less than 10. Allow the user to input positive integers into this matrix. Perform the following tasks on the matrix:

(a) Sort the non-boundary elements in ascending order using any standard sorting technique and rearrange them in the matrix.

(b) Calculate the sum of both the diagonals.

(c) Display the original matrix, rearranged matrix and only the diagonal elements of the rearranged matrix with their sum.

Test your program for the following data and some random data:

Example 1

INPUT: M = 4

```
9   2   1   5
8   13  8   4
15  6   3   11
7   12  23  8
```

OUTPUT:

ORIGINAL MATRIX

```
9   2   1   5
8   13  8   4
15  6   3   11
7   12  23  8
```

REARRANGED MATRIX

```
9   2   1   5
8   3   6   4
15  8   13  11
7   12  23  8
```

DIAGONAL ELEMENTS

```
9           5
    3   6
    8   13
7           8
```

SUM OF THE DIAGONAL ELEMENTS = 59

```java
import java.util.Scanner;

class SortNonBoundary
{
    int m;
    int mat[][];
    SortNonBoundary(int m)
    {
        this.m = m;
        mat = new int[m][m];
    }
    boolean isValid()
    {
        if(m>3 && m<10)
            return true;
        return false;
    }
    void input()
    {
        Scanner sc = new Scanner(System.in);
```

```
        System.out.println("Enter the positive elements: ");
        for(int i=0;i<m;i++)
            for(int j=0;j<m;j++)
                mat[i][j]=sc.nextInt();

    }
    void sort()
    {
        int size = (m-2)*(m-2);
        int arr[] = new int[size];
        int k=0;
        for(int i=1;i<m-1;i++)
        {
            for(int j=1;j<m-1;j++)
            {
                arr[k++] = mat[i][j];
            }
        }
        for(int i=1;i<size;i++)
        {
            for(int j=0;j<size-i;j++)
            {
                if(arr[j]>arr[j+1])
                {
                    int t = arr[j];
                    arr[j] = arr[j+1];
                    arr[j+1] =t;
                }
            }
        }
        k=0;
        for(int i=1;i<m-1;i++)
        {
            for(int j=1;j<m-1;j++)
            {
                mat[i][j]=arr[k++];
            }
        }
    }
    void display()
    {
        for(int i=0;i<m;i++)
        {
            for(int j=0;j<m;j++)
                System.out.print(mat[i][j] + "\t");
            System.out.println();
        }
    }
    void diagonals()
    {
        int s=0;
        for(int i=0;i<m;i++)
        {
            for(int j=0;j<m;j++)
                if(i==j || i+j==m-1)
                {
                    s+=mat[i][j];
                    System.out.print(mat[i][j] + "\t");
                }
                else
                    System.out.print("\t\t");
            System.out.println();
        }
        System.out.println("SUM OF THE DIAOGONAL ELEMENTS = "+s);
    }
```

```
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the size of the matrix: ");
        int m = sc.nextInt();
        SortNonBoundary ob = new SortNonBoundary(m);
        if(!ob.isValid())
            System.out.println("INVALID INPUT");
        else
        {
            ob.input();
            System.out.println("ORIGINAL MATRIX");
            ob.display();
            ob.sort();
            System.out.println("REARRANGED MATRIX");
            ob.display();
            System.out.println("DIAGONAL MATRIX");
            ob.diagonals();
        }
    }
}
```

ROTATE BY 90 DEGREE CLOCKWISE

Write a program to declare a square matrix A[][] of order M x M where 'M' is the number of rows and the number of columns, such that M must be greater than 2 and less than 10. Accept the value of M as user input. Display an appropriate message for an invalid input. Allow the user to input integers into the matrix. Perform the following tasks:

a) Display the original matrix.
b) Rotate the matrix 90° clockwise as shown below:

Original matrix	Rotated matrix
1 2 3	7 4 1
4 5 6	8 5 2
7 8 9	9 6 3

c) Find the sum of the elements of the four corners of the matrix. Test your program for the following data and some random data:

Example 1:

INPUT : M=3

3	4	9
2	5	8
1	6	7

OUTPUT:

ORIGINAL MATRIX

3	4	9
2	5	8
1	6	7

MATRIX AFTER ROTATION

1	2	3
6	5	4
7	8	9

Sum of the corner eements= 20

Example 2:

INPUT : M=4

1	2	4	9
2	5	8	3
1	6	7	4
3	7	6	5

OUTPUT:

ORIGINAL MATRIX

1	2	4	9
2	5	8	3
1	6	7	4
3	7	6	5

MATRIX AFTER ROTATION

3	1	2	1
7	6	5	2
6	7	8	4
5	4	3	9

Sum of the corner eements= 18

EXAMPLE 3.

INPUT : M = 14

OUTPUT:

SIZE OUT OF RANGE

```
import java.util.Scanner;
class RotateClockwise
{
    int m;
    int mat[][];
    RotateClockwise(int m)
    {
        this.m = m;
        mat = new int[m][m];
    }
    boolean isValid()
    {
        if(m>2 && m<10)
            return true;
        return false;
    }
    void input()
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the positive elements: ");
        for(int i=0;i<m;i++)
            for(int j=0;j<m;j++)
                mat[i][j]=sc.nextInt();

    }
    void display()
    {
        for(int i=0;i<m;i++)
        {
            for(int j=0;j<m;j++)
                System.out.print(mat[i][j] + "\t");
```

```
            System.out.println();
        }
    }
    void rotate()
    {
        for(int j=0;j<m;j++)
        {
            for(int i=m-1;i>=0;i--)
                System.out.print(mat[i][j] + " ");
            System.out.println();
        }
    }
    void sumOfCorners()
    {
        int s=mat[0][0] + mat[0][m-1]+mat[m-1][m-1]+mat[m-1][0];
        System.out.println("Sum of the corner elements = " + s);
    }
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the size of the matrix: ");
        int m = sc.nextInt();
        RotateClockwise ob = new RotateClockwise(m);
        if(!ob.isValid())
            System.out.println("OUT OF RANGE");
        else
        {
            ob.input();
            System.out.println("ORIGINAL MATRIX");
            ob.display();
            System.out.println("MATRIX AFTER ROTATION");
            ob.rotate();
            ob.sumOfCorners();
        }
    }
}
```

SYMMETRIC MATRIX

Write a program to declare a square matrix A[][] of order MxM where 'M' is the number of rows and the number of columns, such that M must be greater than 2 and less than 10. Accept the value of M as user input. Display an appropriate message for an invalid input. Allow the user to input integers into this matrix. Perform the following tasks:

a) Display the original matrix.

b) Check if the given matrix is symmetric or not. A square matrix is said to be symmetric, if the element in the i^{th} row and j^{th} column is equal to the element of the j^{th} row and i^{th} column.

c) Find the sum of the elements of left diagonal and the sum of the elements of right diagonal of the matrix and display them.

Test your program for the following data and some random data:

INPUT: M= 3

1	2	3
2	4	5
3	5	6

OUTPUT:

ORIGINAL MATRIX

1	2	3
2	4	5
3	5	6

THE GIVEN MATRIX IS SYMMETRIC

The sum of the left diagonal = 11

The sum of the right diagonal = 10

Example 2

INPUT: M= 4

7	8	9	2
4	5	6	3
8	5	3	1
7	6	4	2

OUTPUT:

ORIGINAL MATRIX

7	8	9	2
4	5	6	3
8	5	3	1
7	6	4	2

THE GIVEN MATRIX IS NOT SYMMETRIC

The sum of the left diagonal = 17

The sum of the right diagonal = 20

Example 3

INPUT: M= 12

OUTPUT:

THE MATRIX SIZE IS OUT OF RANGE

```
import java.util.Scanner;
class SymmetricMatrix
{
    int m;
    int mat[][];
    SymmetricMatrix(int m)
    {
        this.m = m;
        mat = new int[m][m];
```

```
    }
    boolean isValid()
    {
        if(m>2 && m<10)
            return true;
        return false;
    }
    void input()
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the elements: ");
        for(int i=0;i<m;i++)
            for(int j=0;j<m;j++)
                mat[i][j]=sc.nextInt();

    }
    void display()
    {
        for(int i=0;i<m;i++)
        {
            for(int j=0;j<m;j++)
                System.out.print(mat[i][j] + "\t");
            System.out.println();
        }
    }
    void process()
    {
        int sld = 0, srd = 0;
        boolean symmetry = true;
        for(int i=0;i<m;i++)
        {
            sld += mat[i][i];
            srd += mat[i][m-1-i];
            for(int j=0;j<=i;j++)
            {
                if(mat[i][j]!=mat[j][i])
                {
                    symmetry = false;
                }
            }
        }
        if(symmetry)
            System.out.println("THE GIVEN MATRIX IS SYMMETRIC");
        else
            System.out.println("THE GIVEN MATRIX IS NOT SYMMETRIC");
        System.out.println("The sum of left diagonal = " + sld);
        System.out.println("the sum of right diagonal = " + srd);
    }
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the size of the matrix: ");
        int m = sc.nextInt();
        SymmetricMatrix ob = new SymmetricMatrix(m);
        if(!ob.isValid())
            System.out.println("OUT OF RANGE");
        else
        {
            ob.input();
            System.out.println("ORIGINAL MATRIX");
            ob.display();
            ob.process();
        }
    }
}
```

WONDROUS SQUARE

Question 2.

A wondrous square is an n by n grid which fulfils the following conditions:
1) It contains integers from 1 to n2, where each integer appears only once.
2) The sum of integers in any row or column must add up to 0.5 x n x (n^2 + 1).

For e.g. the following grid is a wondrous square where the sum of each row or column is 65 when n=5:

17	24	1	8	15
23	5	7	14	16
4	6	13	20	22
10	12	19	21	3
11	18	25	2	9

Write a program to read n (2 <= n <= 10) and the values stored in these n by n cells and output if the grid represents a wondrous square or not. Also output all the prime numbers in the grid along with their row index and column index as shown in the output. A natural number is said to be prime if it has exactly two divisors. E.g. 2,3,5,7,11.....

The first element of the given grid i.e. 17 is stored at row index 0 and column index 0 and the next element in the row i.e. 24 is stored at row index 0 and column index 1.

Test your program for the following data and some random data.
Sample Data:
Input:
N=4

16	15	1	2
6	4	10	14
9	8	12	5
3	7	11	13

Output:

Yes it represents a wondrous square.

Prime	Row Index	Column Index
2	0	3
3	3	0
5	2	3
7	3	1
11	3	2
13	3	3

Input:
N=3

1	2	4
3	7	5
8	9	6

Output:
Not a wondrous square.

Prime	Row Index	Column Index
2	0	0
2	1	1

```
import java.util.Scanner;
class WondrousSquare
{
    int n;
    int mat[][];
    WondrousSquare(int n)
    {
        this.n = n;
        mat = new int[n][n];
    }
    boolean isValid()
    {
        if(n>=2 && n<=10)
            return true;
        return false;
    }
```

```
    void input()
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the elements: ");
        for(int i=0;i<n;i++)
            for(int j=0;j<n;j++)
                mat[i][j]=sc.nextInt();

    }
    void display()
    {
        for(int i=0;i<n;i++)
        {
            for(int j=0;j<n;j++)
                System.out.print(mat[i][j] + "\t");
            System.out.println();
        }
    }
    boolean isPrime(int n)
    {
        int c=0;
        for(int i=1;i<=n;i++)
            if (n%i==0)
                c++;
        return c==2;
    }
    boolean isWondrous()
    {
        boolean allpresent = true;
        outerloop: for(int k=1;k<=n*n;k++)
        {
            for(int i=0;i<n;i++)
            {
                for(int j=0;j<n;j++)
                {
                    if(k == mat[i][j])
                    {
                        continue outerloop;
                    }
                }
            }
            allpresent = false;
            break;
        }

        if(!allpresent)
            return false;

        int s = n * (n*n + 1) / 2;
        boolean wondrous =true;
        for(int i=0;i<n;i++)
        {
            int sr = 0,sc = 0;
            for(int j=0;j<n;j++)
            {
                sr += mat[i][j];
                sc += mat[j][i];
            }
            if( s != sr || s != sc)
            {
                wondrous = false;
                break;
            }
        }
        return wondrous;
```

```
    }
    void printPrimes()
    {
        System.out.printf("%-20s%-20s%-20s\n","Prime","Row Index","Column Index");
        for(int i=0;i<n;i++)
        {
            for(int j=0;j<n;j++)
            {
                if(isPrime(mat[i][j]))
                    System.out.printf("%-20d%-20d%-20d\n",mat[i][j],i,j);
            }
        }
    }
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the size of the matrix: ");
        int n = sc.nextInt();
        WondrousSquare ob = new WondrousSquare(n);
        if(!ob.isValid())
            System.out.println("OUT OF RANGE");
        else
        {
            ob.input();
            System.out.println("ORIGINAL MATRIX");
            ob.display();
            if(ob.isWondrous())
                System.out.println("Yes, it represents a wondrous square");
            else
                System.out.println("Not a wondrous square");
            ob.printPrimes();
        }
    }
}
```

MIRROR IMAGE

Write a program to declare a square matrix A[][] of order (M x M) where M is the number of rows and the number of columns such that M must be greater than 2 and less than 20. Allow the user to input integers into this matrix. Display the appropriate error message for an invalid input. Perform the following tasks:

a) Display the input matrix

b) Create a mirror image of the inputted matrix

c) Display the mirror image matrix

Test your program for the following data and some random data:
Example 1

INPUT : M = 3

4	16	12
8	2	14
6	1	3

OUTPUT :

ORIGINAL MATRIX

4	16	12
8	2	14
6	1	3

MIRROR IMAGE MATRIX

12	16	4
14	2	8
3	1	6

Example 2

INPUT : M=22

OUTPUT : SIZE OUT OF RANGE

```
import java.util.Scanner;
class MirrorImage
{
    int m;
    int mat[][];
    int image[][];
    MirrorImage(int m)
    {
        this.m = m;
        mat = new int[m][m];
        image = new int[m][m];
    }
    boolean isValid()
    {
        if(m>2 && m<20)
            return true;
```

```
        return false;
    }
    void input()
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the elements: ");
        for(int i=0;i<m;i++)
            for(int j=0;j<m;j++)
                mat[i][j]=sc.nextInt();

    }
    void display()
    {
        for(int i=0;i<m;i++)
        {
            for(int j=0;j<m;j++)
                System.out.print(mat[i][j] + "\t");
            System.out.println();
        }
    }
    void convert()
    {
        for(int i=0;i<m;i++)
        {
            for(int j=0;j<m;j++)
                image[i][m-1-j]=mat[i][j];
        }
    }
    void displayMirrorImage()
    {
        for(int i=0;i<m;i++)
        {
            for(int j=0;j<m;j++)
                System.out.print(image[i][j] + "\t");
            System.out.println();
        }
    }
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the size of the matrix: ");
        int m = sc.nextInt();
        MirrorImage ob = new MirrorImage(m);
        if(!ob.isValid())
            System.out.println("SIZE OUT OF MATRIX");
        else
        {
            ob.input();
            System.out.println("ORIGINAL MATRIX");
            ob.display();
            ob.convert();
            System.out.println("Mirror Image Matrix: ");
            ob.displayMirrorImage();
        }
    }
}
```

MAX AND MIN WITH POSITION

Question 3.

Write a program to declare a matrix A[][] of order (M x N) where 'M' is the number of rows and 'N' is the number of columns such that both M and N must be greater than 2 and less than 20. Allow the user to input integers into this matrix. Perform the following tasks on the matrix:

a) Display the input matrix.
b) Find the maximum and minimum value in the matrix and display them along with their position.
c) Sort the elements of the matrix in ascending order using any standard sorting technique and rearrange them in the matrix.
d) Output the rearranged matrix.

Test your program for the following data and some random data:

Example 1

INPUT M= 3 N= 4

8	7	9	3
-2	0	4	5
1	3	6	-4

OUTPUT :

ORIGINAL MATRIX

8	7	9	3
-2	0	4	5
1	3	6	-4

LARGEST NUMBER : 9 ROW = 0 COLUMN = 2

SMALLEST NUMBER : -4 ROW = 2 COLUMN = 3

REARRANGED MATRIX

-4	-2	0	1
3	3	4	5
6	7	8	9

EXAMPLE 2

INPUT M= 3 N = 22

OUTPUT:

SIZE OUT OF RANGE

```java
import java.util.Scanner;
class MaxMinPosition
{
    int m,n;
    int mat[][];
    MaxMinPosition(int m, int n)
    {
        this.m = m;
        this.n = n;
        mat = new int[m][n];
    }
    boolean isValid()
    {
        if(m>2 && m<20 && n>2 && n<20)
            return true;
        return false;
    }
    void input()
    {
        Scanner sc = new Scanner(System.in);
```

```
        System.out.println("Enter the elements: ");
        for(int i=0;i<m;i++)
            for(int j=0;j<n;j++)
                mat[i][j]=sc.nextInt();

    }
    void display()
    {
        for(int i=0;i<m;i++)
        {
            for(int j=0;j<n;j++)
                System.out.print(mat[i][j] + "\t");
            System.out.println();
        }
    }
    void findMaxMin()
    {
        int maxi, maxj, mini, minj;
        maxi=maxj=mini=minj=0;
        for(int i=0;i<m;i++)
        {
            for(int j=0;j<n;j++)
            {
                if(mat[i][j]>mat[maxi][maxj])
                {
                    maxi=i;
                    maxj=j;
                }
                else if(mat[i][j]<mat[mini][minj])
                {
                    mini=i;
                    minj=j;
                }
            }
        }
        System.out.println("Largest Number: " + mat[maxi][maxj]);
        System.out.println("Row : " + maxi);
        System.out.println("Column: " + maxj);
        System.out.println("Smallest Number: " + mat[mini][minj]);
        System.out.println("Row : "+ mini);
        System.out.println("Column: " + minj);
    }
        void sort()
    {
        int size = m*n;
        int arr[] = new int[size];
        int k=0;
        for(int i=0;i<m;i++)
        {
            for(int j=0;j<n;j++)
            {
                arr[k++] = mat[i][j];
            }
        }
        for(int i=1;i<size;i++)
        {
            for(int j=0;j<size-i;j++)
            {
                if(arr[j]>arr[j+1])
                {
                    int t = arr[j];
                    arr[j] = arr[j+1];
                    arr[j+1] =t;
                }
            }
```

```
        }
        k=0;
        for(int i=0;i<m;i++)
        {
            for(int j=0;j<n;j++)
            {
                mat[i][j]=arr[k++];
            }
        }
    }
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the size of the matrix: ");
        int m = sc.nextInt();
        int n = sc.nextInt();
        MaxMinPosition ob = new MaxMinPosition(m,n);
        if(!ob.isValid())
            System.out.println("SIZE OUT OF MATRIX");
        else
        {
            ob.input();
            System.out.println("ORIGINAL MATRIX");
            ob.display();
            ob.findMaxMin();
            ob.sort();
            ob.display();

        }
    }
}
```

STRING ARRAY ENCRYPTION

Question 2.

Encryption is a technique of coding messages to maintain their secrecy. A string of size 'n' where n is greater than 1 and less than 10, stores single sentences(each sentence ends with a full stop) in each row of the array.

Write a program to accept the size of the array . Display an appropriate message if the size is not satisfying the given condition. Define a string array of the input size and fill it with sentences row-wise.

Change the sentences of the odd rows with an encryption of two characters ahead of the original characters. Also change the sentences of the even rows by storing the sentence in reverse order. Display the encrypted sentences as per the sample data given below:

Test your program with some sample data and some random data.
Example 1.
INPUT: n = 4
IT IS CLOUDY.
IT MAY RAIN.
THE WEATHER IS FINE.
IT IS COOL.

OUTPUT:
KV KU ENQWFA.
RAIN MAY IT.
VJG YGCVJGT KU HKPG.
COOL IS IT.

```
import java.util.Scanner;
import java.util.StringTokenizer;
class StringArray
{
    int n;
    String arr[];
    StringArray(int n)
    {
        this.n = n;
        arr = new String[n];
    }
    boolean isValid()
    {
        if(n>1 && n<10)
            return true;
        return false;
    }
    void input()
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the strings: ");
        for(int i=0;i<n;i++)
            arr[i]= sc.nextLine();
    }
    String moveAhead(String w)
    {
        String r = "";
        for(int i=0;i<w.length();i++)
        {
            char ch = w.charAt(i);
            ch+=2;
            if(ch>90)
                ch-=26;
            r += ch;
        }
```

```
        return r;
    }
    void encrypt()
    {
        for(int i=0;i<n;i++)
        {

            String result="";
            StringTokenizer st = new
StringTokenizer(arr[i].substring(0,arr[i].length()-1));
            while(st.hasMoreTokens())
            {
                String word = st.nextToken();
                if(i%2==0)
                    result += moveAhead(word) + " ";
                else
                    result = word + " " + result;
            }
            arr[i] = result.substring(0,result.length()-1) + ".";
        }
    }
    void display()
    {
        System.out.println("After Encryption: ");
        for(int i=0;i<n;i++)
        {
            System.out.println(arr[i]);
        }
    }
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the value for n: ");
        int n = Integer.parseInt(sc.nextLine());
        StringArray ob = new StringArray(n);
        if(!ob.isValid())
            System.out.println("INVALID ENTRY");
        else
        {
            ob.input();
            ob.encrypt();
            ob.display();
        }
    }
}
```

PENDULUM SORTING

Write a program to input a list of integers in an array. Now, perform the following tasks:

(a) Arrange the array elements in an ascending order of their values.

(b) Arrange them in a way similar to the to-and-fro movement of a pendulum. It means the lowest element out of the list of integers, must come in center position of array. The number in the ascending order next to the lower, goes to the left, the next higher number goes to the right of lower number and it continues. As higher numbers are reached, one goes to either side of the lower value in a to-and-fro manner similar to that of a pendulum.

Sample data:

Example 1:

```
Enter number of elements: 5
Enter Element 1: 10
Enter Element 2: 22
Enter Element 3: 31
Enter Element 4: 45
Enter Element 5: 53
The Sorted Array is:
10  22  31  45  53
The Result is:
45  22  10  31  53
```

Example 2:

```
Enter number of elements: 7
Enter Element 1: 12
Enter Element 2: 34
Enter Element 3: 11
Enter Element 4: 5
Enter Element 5: 67
Enter Element 6: 20
Enter Element 7: 47
The Sorted Array is:
5  11  12  20  34  47  67
The Result is:
47  20  11  5  12  34  67
```

```java
import java.util.Scanner;

class PendulumSorting
{
    int n;
    int arr[];
    PendulumSorting()
    {

    }
    void input()
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the number of elements: ");
        n = sc.nextInt();
        arr = new int[n];
        for(int i=0;i<n;i++)
        {
            System.out.print("Enter Element "+(i+1)+": ");
            arr[i] =sc.nextInt();
        }
```

```
    }
    void sort()
    {
        for(int i=1;i<n;i++)
        {
            for(int j=0;j<n-i;j++)
            {
                if(arr[j]>arr[j+1])
                {
                    int t = arr[j];
                    arr[j] = arr[j+1];
                    arr[j+1] = t;
                }
            }
        }

        System.out.println("\nThe sorted array is: ");
        display();
    }
    void display()
    {
        for(int i=0;i<n;i++)
        {
            System.out.print(arr[i] + " " );
        }
    }
    void pendulum()
    {
        int mid = n/2;
        int arr2[] = new int[n];
        int k=0;
        arr2[mid] = arr[k++];
        for(int i=1;i<=mid;i++)
        {
            arr2[mid-i] = arr[k++];
            arr2[mid+i] = arr[k++];
        }
        arr = arr2;
        System.out.println("\nThe result is: ");
        display();
    }
    public static void main(String args[])
    {
        PendulumSorting ob = new PendulumSorting();
        ob.input();
        ob.sort();
        ob.pendulum();
    }
}
```