ABOUT JAVA

**Java** is a general-purpose computer programming language that is concurrent, class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of computer architecture. As of 2016, Java is one of the most popular programming languages in use, particularly for client-server web applications, with a reported 9 million developers.Java was originally developed by James Gosling at Sun Microsystems (which has since been acquired by Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++, but it has fewer low-level facilities than either of them.

The original and reference implementation Java compilers, virtual machines, and class libraries were originally released by Sun under proprietary licenses. As of May 2007, in compliance with the specifications of the Java Community Process, Sun relicensed most of its Java technologies under the GNU General Public License. Others have also developed alternative implementations of these Sun technologies, such as the GNU Compiler for Java (bytecode compiler), GNU Classpath (standard libraries), and IcedTea-Web (browser plugin for applets).

The latest version is Java 9, released on September 21, 2017, and is one of the two versions currently supported for free by Oracle. Versions earlier than Java 8 are supported by companies on a commercial basis.

ABOUT BLUE J

**BlueJ** is an integrated development environment (IDE) for the Java programming language, developed mainly for educational purposes, but also suitable for small-scale software development. It runs with the help of JDK (Java Development Kit).

BlueJ was developed to support the learning and teaching of object-oriented programming, and its design differs from other development environments as a result. The main screen graphically shows the class structure of an application under development (in a UML-like diagram), and objects can be interactively created and tested. This interaction facility, combined with a clean, simple user interface, allows easy experimentation with objects under development. Object-oriented concepts (classes, objects, communication through method calls) are represented visually and in its interaction design in the interface.

***JAVA PROGRAMS***

**Question 1:**

Write a program to input a word from the user and remove the duplicate characters present in it.

**Example:**

INPUT – abcabcabc  
OUTPUT – abc

INPUT – javaforschool  
OUTPUT – javforschl

import java.io.\*;

class RemoveDupChar

{

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

    {

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

        System.out.print("Enter any word : ");

        String s = br.readLine();

        int l = s.length();

        char ch;

        String ans="";

        for(int i=0; i<l; i++)

        {

            ch = s.charAt(i);

            if(ch!=' ')

                ans = ans + ch;

            s = s.replace(ch,' '); //Replacing all occurrence of the current character by a space

        }

       System.out.println("Word after removing duplicate characters : " + ans);

    }

#### }

#### Output:

**Example 1:**  
Enter any word : Mississippi  
Word after removing duplicate characters : Misp

#### ****Question 2:****

Write a Program in Java to input a 2-D square matrix and check whether it is a Scalar Matrix or not.

**Scalar Matrix :** A scalar matrix is a **diagonal matrix** in which the main diagonal () entries are all equal.

import java.util.\*;

class ScalarMatrix

{

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

    {

        Scanner sc=new Scanner(System.in);

        System.out.print("Enter the size of the matrix : ");

        int m=sc.nextInt();

        int A[][]=new int[m][m];

        /\* Inputting the matrix \*/

        for(int i=0;i<m;i++)

        {

            for(int j=0;j<m;j++)

            {

                System.out.print("Enter an element : ");

                A[i][j]=sc.nextInt();

            }

        }

        /\* Printing the matrix \*/

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

        System.out.println("The Matrix is : ");

        for(int i=0;i<m;i++)

        {

            for(int j=0;j<m;j++)

            {

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

            }

            System.out.println();

        }

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

        int p = 0, q = 0, x = A[0][0]; // 'x' is storing the 1st main diagonal element

        for(int i=0;i<m;i++)

        {

            for(int j=0;j<m;j++)

            {

                /\* Checking that the matrix is diagonal or not \*/

                if(i!=j && A[i][j]!=0) // All non-diagonal elements must be zero

                {

                    p=1;

                    break;

                }

                /\* Checking the matrix for scalarity \*/

                // All main diagonal elements must be equal to 'x' and non-zero

                if(i==j && (A[i][j]==0 || A[i][j]!=x))

                {

                    q=1;

                    break;

                }

            }

        }

        if(p==0 && q==0)

            System.out.println("The matrix is scalar");

        else

            System.out.println("The matrix is not scalar");

    }

}

#### ****Output:****

Enter the size of the matrix : 4  
Enter an element : 5  
Enter an element : 0  
Enter an element : 0  
Enter an element : 0  
Enter an element : 0  
Enter an element : 5  
Enter an element : 0  
Enter an element : 0  
Enter an element : 0  
Enter an element : 0  
Enter an element : 5  
Enter an element : 0  
Enter an element : 0  
Enter an element : 0  
Enter an element : 0  
Enter an element : 5  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
The Matrix is :  
5 0 0 0  
0 5 0 0  
0 0 5 0  
0 0 0 5  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
The matrix is Scalar

**Question 3:**

A sentence in the Special Fashion can be printed by taking two integers (not beyond total number of words in the sentence or less than 1). These integers tell the word number of the sentence. Replace only those words present at those given integer places by the next character in a circular fashion according to the English Alphabets. If both the integers are same then replace only one word. let us consider the following examples:

**Input Sentence:** He has good Books.  
**Input Integers:** 2, 4  
**Output Sentence:**He ibt good Cpplt.  
(i.e. word number 2 and 4 have been replaced by the next characters in a circular fashion)

**Input Sentence:** Time and tide waits for none.  
**Input Integers:** 3, 3  
**Output Sentence:**Time and ujef waits for none.

Write a case sensitive program that reads a sentence from console (the characters of the sentence may be capital or small or mixed) and two positive integers and output the same sentence after replacing those words present at those given integer places by the next character in a circular fashion according to the English Alphabets.

In the first example given above, word number 2, i.e. “has” is replaced by next characters and hence it becomes “ibt”. Similarly, word number 4, i.e. “Books” is replaced by next characters and hence it becomes “Cpplt”.

import java.io.\*;

class Special\_Fashion

{

    String repChar(String s) //function doing the work of replacing characters

    {

        int l = s.length(), a = 0;

        char ch;

        String res="";

        for(int i=0; i<l; i++)

        {

            ch = s.charAt(i); //extracting characters one by one

            a = ch + 1; //storing ASCII values after adding 1 to the current character

            if(ch=='z' || ch=='Z')

            {

                a = a - 26;

            }

            res = res + (char)a; //finally adding the changed character to the new String

        }

        return res;

    }

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

    {

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

        Special\_Fashion ob = new Special\_Fashion();

        System.out.print("nEnter any sentence : "); //Inputting the sentence

        String s = br.readLine();

        String ans=""; //String variable to store the final result

        String word[]=s.split("[. ]+"); //saving the words of the sentence in an array using split()

        int c = word.length;

        System.out.print("Enter the 1st word number : ");

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

        System.out.print("Enter the 2nd word number : ");

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

        if(x<1 || y<1 || x>c || y>c) //checking whether integers inputted are acceptable or not

        {

            System.out.println("Sorry! The word numbers inputted are out of range");

        }

        else

        {

                if(x != y)

                {

                    word[y-1]=ob.repChar(word[y-1]); //sending the words to the repChar() function

                }

                word[x-1]=ob.repChar(word[x-1]);

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

                {

                    ans = ans + word[i] + " ";

                }

                System.out.print("Output = "+ans.trim()+".");

        }

    }

}

#### ****Output:****

**1)**Enter any sentence : I love Java for School.  
Enter the 1st word number : 2  
Enter the 2nd word number : 5  
Output = I mpwf Java for Tdippm.

**2)**Enter any sentence : I love Java for School  
Enter the 1st word number : 4  
Enter the 2nd word number : 4  
Output = I love Java gps School.

**3)**Enter any sentence : I love Java for School  
Enter the 1st word number : 2  
Enter the 2nd word number : 6  
Sorry! The word numbers inputted are out of range

**Question 4:**

The encryption of alphabets are 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.\*;

class WordPotential

{

    int findPotential(String s) // Function to find potential of a word

    {

        s = s.toUpperCase();

        int p = 0, l = s.length();

        char ch;

        for(int i=0; i<l; i++)

        {

            ch = s.charAt(i);

            p = p + (ch-64); // if ch = 'A', then 'A'-64 = ASCII value of 'A' - 64 = 65-64 = 1

        }

        return p;

    }

    // Function to sort the words in ascending order of their potential

    void sortPotential(String w[], int p[])

    {

        int n = w.length, t1 = 0;

        String t2 = "";

        for(int i=0; i<n-1; i++)

        {

            for(int j=i+1; j<n; j++)

            {

                if(p[i]>p[j])

                {

                    t1 = p[i];

                    p[i] = p[j];

                    p[j] = t1;

                    t2 = w[i];

                    w[i] = w[j];

                    w[j] = t2;

                }

            }

        }

        printResult(w,p);

    }

    void printResult(String w[], int p[]) // Function to print the final result

    {

        int n = w.length;

        String ans = "";

        for(int i=0; i<n; i++)

        {

            ans = ans + " " + w[i];

        }

        ans = ans.trim();

        System.out.println("\nOutput\t\t :  \t"+ans);

    }

    public static void main(String args[])

    {

        WordPotential ob = new WordPotential();

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter a sentence : \t");

        String s = sc.nextLine();

        StringTokenizer str = new StringTokenizer(s," .,?!");

        int n = str.countTokens();

        String words[] = new String[n];

        int potential[] = new int[n];

        for(int i=0; i<n; i++)

        {

            words[i] = str.nextToken(); // Saving words one by one in an array

            potential[i] = ob.findPotential(words[i]); // Saving potential of every word

        }

        // Printing the words along with their potential

        System.out.print("\nPotential\t : \t");

        for(int i=0; i<n; i++)

        {

            System.out.println(words[i]+"\t= "+potential[i]);

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

        }

        ob.sortPotential(words,potential);

    }

}

**Output:**

|  |
| --- |
| Enter a sentence :  Look before you leap.    Potential        :  Look    = 53                      before  = 51                      you = 61                      leap    = 34    Output       :      leap before Look you      Enter a sentence :  The sky is the limit.    Potential        :  The = 33                      sky = 55                      is  = 28                      the = 33                      limit   = 63    Output       :      is The the sky limit |

#### ****Question 5:****

The MOBIUS function M(N) for a natural number N is defined as follows:

M(N) = 1                          if N = 1 [Condition 1]

M(N) = 0                          if  any prime factor of N is contained in N more than once [Condition 2]

M(N) = (-1)p                    if N is a product of ‘p’ distinct prime factors [Condition 3]

**Example :**

M(78) = -1                ( for 78 = 2 \* 3 \* 13     M(78) = ( -1)3 = -1 )

M(34) = 1                 ( for 34 = 2 \* 17           M(34) = ( -1)2 = 1 )

M(12) = 0                 ( for 12 = 2 \* 2 \* 3       M(12) = 0 for 2 appears two times)

M(17) = -1                ( for 17 = 17                 M(17) = ( -1)1 = -1 )

Design a class MobiusFn to define Mobius function for a natural number n.

**Class name :** MobiusFn

**Data members/Instance variables:**  
n : stores an integer number

**Member functions:**  
MobiusFn() : default constructor  
void input() : input value of n  
int primeFac() : to check and count prime factors of n  
void display() : to find and print values of Mobius function

Specify the class **MobiusFn** giving details of the constructors, **void input()**, **int primeFac()**, and **void display()**. Also define the main function to create an object and call methods accordingly to enable the task.

import java.util.\*;

class MobiusFn

{

    int n;

    MobiusFn()

    {

        n = 0;

    }

    void input()

    {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter a number : ");

        n = sc.nextInt();

    }

    /\*  The function primefac() either returns '0' if prime factors are repeated

     \*  or returns the no.of prime factors \*/

    int primeFac()

    {

        int a=n, i=2, m=0, c=0, f=0;

        while(a > 1) // loop to generate prime factors

        {

            c = 0; // variable to store frequency of every prime factor

            while(a%i == 0) // if 'i' is a prime factor

            {

                c++; // counting frequency of 'i'

                f++; // counting no of prime factors

                a=a/i;

            }

                i++;

            if(c > 1) // returning '0' if prime factors are repeated

                return 0;

        }

        return f; // returning no. of prime factors

    }

    void display() // function to display value of mobius function

    {

        int mob,x;

        if(n == 1) // condition 1

            mob = 1;

        else

        {

            x = primeFac();

            if(x == 0) // condition 2

                mob = 0;

            else // condition 3

                mob = (int)Math.pow(-1,x);

        }

        System.out.println("Value of Mobius Function : "+mob);

    }

    public static void main(String args[])

    {

        MobiusFn ob = new MobiusFn();

        ob.input();

        ob.display();

    }

}

**Output:**

|  |
| --- |
| Enter a number : 78  Value of Mobius Function : -1    Enter a number : 12  Value of Mobius Function : 0    Enter a number : 34  Value of Mobius Function : 1    Enter a number : 17  Value of Mobius Function : -1 |

#### ****Question6:****

Write a Program in Java to input a number and check whether it is a **Pronic Number** or **Heteromecic Number** or not.

**Pronic Number :** A pronic number, oblong number, rectangular number or heteromecic number, is a number which is the product of two consecutive integers, that is, n (n + 1).

The first few pronic numbers are:  
0, 2, 6, 12, 20, 30, 42, 56, 72, 90, 110, 132, 156, 182, 210, 240, 272, 306, 342, 380, 420, 462 … etc.

import java.util.\*;

class PronicNumber

{

    public static void main(String args[])

    {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter a number : ");

        int n = sc.nextInt();

        int flag = 0;

        for(int i=0; i<n; i++)

        {

            if(i\*(i+1) == n)

            {

                flag = 1;

                break;

            }

        }

        if(flag == 1)

            System.out.println(n+" is a Pronic Number.");

        else

            System.out.println(n+" is not a Pronic Number.");

    }

}

**Output:**

|  |
| --- |
| Enter a number : 110  110 is a Pronic Number.    Enter a number : 73  73 is not a Pronic Number.    Enter a number : 342  342 is a Pronic Number.    Enter a number : 15  15 is not a Pronic Number. |

#### ****Question7:****

Write a Program in Java to input a number and check whether it is a **Harshad Number** or **Niven Number** or not..

**Harshad Number :** In recreational mathematics, a Harshad number (or Niven number), is an integer (in base 10) that is divisible by the sum of its digits.

Let’s understand the concept of Harshad Number through the following example:

* The number 18 is a Harshad number in base 10, because the sum of the digits 1 and 8 is 9 (1 + 8 = 9), and 18 is divisible by 9 (since 18 % 9 = 0)
* The number 1729 is a Harshad number in base 10, because the sum of the digits 1 ,7, 2 and 9 is 19 (1 + 7 + 2 + 9 = 19), and 1729 is divisible by 19 (1729 = 19 \* 91)
* The number 19 is not a Harshad number in base 10, because the sum of the digits 1 and 9 is 10 (1 + 9 = 10), and 19 is not divisible by 10 (since 19 % 10 = 9)

The first few Harshad numbers in base 10 are:  
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 18, 20, 21, 24, 27, 30, 36, 40, 42, 45, 48, 50, 54, 60, 63, 70, 72, 80, 81, 84, 90, 100, 102, 108, 110, 111, 112, 114, 117, 120, 126, 132, 133, 135, 140, 144, 150, 152, 153, 156, 162, 171, 180, 190, 192, 195, 198, 200 etc.

import java.util.\*;

class HarshadNumber

{

    public static void main(String args[])

    {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter a number : ");

        int n = sc.nextInt();

        int c = n, d, sum = 0;

        //finding sum of digits

        while(c>0)

        {

            d = c%10;

            sum = sum + d;

            c = c/10;

        }

        if(n%sum == 0)

            System.out.println(n+" is a Harshad Number.");

        else

            System.out.println(n+" is not a Harshad Number.");

    }

}

**Output:**

|  |
| --- |
| Enter a number : 195  195 is a Harshad Number.    Enter a number : 194  194 is not a Harshad Number.    Enter a number : 190  190 is a Harshad Number.    Enter a number : 111  111 is a Harshad Number. |

#### ****Question8:****

Write a Program in Java to input a number and check whether it is a **Fascinating Number** or not..

**Fascinating Numbers :** Some numbers of 3 digits or more exhibit a very interesting property. The property is such that, when the number is multiplied by 2 and 3, and both these products are concatenated with the original number, all digits from 1 to 9 are present exactly once, regardless of the number of zeroes.

Let’s understand the concept of Fascinating Number through the following example:

Consider the number 192,  
192 x 1 = 192  
192 x 2 = 384  
192 x 3 = 576

Concatenating the results : **192384576**

It could be observed that ‘192384576’ consists of all digits from 1 to 9 exactly once. Hence, it could be concluded that 192 is a Fascinating Number.

Some examples of fascinating Numbers are : 192, 219, 273, 327, 1902, 1920, 2019 etc.

import java.util.\*;

class FascinatingNumber

{

    boolean isUnique(String q)

    {

        int A[] = {0,0,0,0,0,0,0,0,0,0}; //to store frequency of every digit from '0' to '9'

        int i, flag = 0;

        char ch;

        for(i=0; i<q.length(); i++)

        {

            ch = q.charAt(i);

            A[ch-48]++;

            /\*  increasing A[5] if ch='5' as '5'-48 = 53-48=5

             \*  (ASCII values of '0' to '9' are 48 to 57) \*/

        }

        for(i=1; i<10; i++)

        {

            //checking if every digit from '1' to '9' are present exactly once or not

            if(A[i]!=1)

            {

                flag = 1; //flag is set to 1 if frequency is not 1

                break;

            }

        }

        if(flag == 1)

            return false;

        else

            return true;

    }

    public static void main(String args[])

    {

        Scanner sc = new Scanner(System.in);

        FascinatingNumber ob = new FascinatingNumber();

        System.out.print("Enter a number : ");

        int n = sc.nextInt();

        String p = Integer.toString(n); //converting the number to String

        if(p.length()<3)

            System.out.println("Number should be of atleast 3 digits.");

        else

        {

            String s = Integer.toString(n\*1) + Integer.toString(n\*2) + Integer.toString(n\*3);

            /\*  Joining the first, second and third multiple of the number

             \*  by converting them to Strings and concatenating them\*/

            if(ob.isUnique(s))

                System.out.println(n+" is a Fascinating Number.");

            else

                System.out.println(n+" is not a Fascinating Number.");

        }

    }

}

**Output:**

|  |
| --- |
| Enter a number : 273  273 is a Fascinating Number.    Enter a number : 853  853 is not a Fascinating Number.    Enter a number : 95  Number should be of atleast 3 digits. |

#### ****Question9:****

Write a Program in Java to input a number and check whether it is an **Evil Number** or not.

**Evil Number :** An Evil number is a positive whole number which has even number of 1’s in its binary equivalent.

**Example:** Binary equivalent of 9 is 1001, which contains even number of 1’s.  
A few evil numbers are 3, 5, 6, 9….

Design a program to accept a positive whole number and find the binary equivalent of the number and count the number of 1’s in it and display whether it is a Evil number or not with an appropriate message. Output the result in format given below:

**Example 1**  
INPUT : 15  
BINARY EQUIVALENT : 1111  
NO. OF 1’s : 4  
OUTPUT : EVIL NUMBER

**Example 2**  
INPUT : 26  
BINARY EQUIVALENT : 11010  
NO. OF 1’s : 3  
OUTPUT : NOT AN EVIL NUMBER

import java.util.\*;

class EvilNumber

{

    String toBinary(int n) // Function to convert a number to Binary

    {

        int r;

        String s=""; //variable for storing the result

        char dig[]={'0','1'}; //array storing the digits (as characters) in a binary number system

        while(n>0)

            {

                r=n%2; //finding remainder by dividing the number by 2

                s=dig[r]+s; //adding the remainder to the result and reversing at the same time

                n=n/2;

            }

        return s;

    }

    int countOne(String s) // Function to count no of 1's in binary number

    {

        int c = 0, l = s.length();

        char ch;

        for(int i=0; i<l; i++)

        {

            ch=s.charAt(i);

            if(ch=='1')

            {

                c++;

            }

        }

        return c;

    }

    public static void main(String args[])

    {

        EvilNumber ob = new EvilNumber();

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter a positive number : ");

        int n = sc.nextInt();

        String bin = ob.toBinary(n);

        System.out.println("Binary Equivalent = "+bin);

        int x = ob.countOne(bin);

        System.out.println("Number of Ones = "+x);

        if(x%2==0)

            System.out.println(n+" is an Evil Number.");

        else

            System.out.println(n+" is Not an Evil Number.");

    }

}

**Output:**

|  |
| --- |
| Enter a positive number : 26  Binary Equivalent = 11010  Number of Ones = 3  26 is Not an Evil Number.    Enter a positive number : 420  Binary Equivalent = 110100100  Number of Ones = 4  420 is an Evil Number.    Enter a positive number : 659  Binary Equivalent = 1010010011  Number of Ones = 5  659 is Not an Evil Number. |

#### ****Question10:****

Write a Program in Java to input a word and print its anagrams..

**Note:** Anagrams are words made up of all the characters present in the original word by re-arranging the characters.

**Example:** Anagrams of the word TOP are: TOP, TPO, OPT, OTP, PTO and POT

import java.util.\*;

class Anagrams

{

    int c = 0;

    void input()throws Exception

    {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter a word : ");

        String s = sc.next();

        System.out.println("The Anagrams are : ");

        display("",s);

        System.out.println("Total Number of Anagrams = "+c);

    }

    void display(String s1, String s2)

    {

        if(s2.length()<=1)

        {

            c++;

            System.out.println(s1+s2);

        }

        else

        {

            for(int i=0; i<s2.length(); i++)

            {

                String x = s2.substring(i, i+1);

                String y = s2.substring(0, i);

                String z = s2.substring(i+1);

                display(s1+x, y+z);

            }

        }

    }

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

    {

        Anagrams ob=new Anagrams();

        ob.input();

    }

}

#### Question 11:

A class Admission contain the admission numbers of 100 students. Some of the data members/ member functions are given below:

**Class name:** Admission

**Data member/instance variable:**

Adno[ ]: Integer array to store admission numbers

**Member functions/methods:**

Admission(): constructur to initialize the array elements  
void fillArray(): to accept the element of the array in ascending order  
int binSearch(int l, int u, int v): to search for a particular admission number(v) using binary search and recursive technique and return 1 if found otherwise returns -1

Specify the class **Admission** giving details of the**constructor**, **void fillArrray()** and **int binSearch(int, int, int)**. Define the **main()** function to create an object and call the functions accordingly to enable task.

import java.util.\*;

class Admission

{

    int Adno[]=new int[100];

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

    Admission() // Default constructor

    {

        for(int i=0; i<100; i++)

        {

            Adno[i]=0;

        }

    }

    void fillArray()throws Exception // Function to accept elements in ascending order

    {

        for(int i=0; i<100; i++)

        {

            System.out.print("Enter Admission no of student "+(i+1)+": ");

            Adno[i] = sc.nextInt();

        }

        /\*Sorting the array in ascending order \*/

        int temp=0;

        for(int i=0; i<99; i++)

        {

            for(int j=i+1; j<100; j++)

            {

                if(Adno[i]>Adno[j])

                {

                    temp = Adno[i];

                    Adno[i] = Adno[j];

                    Adno[j] = temp;

                }

            }

        }

    }

    int binSearch(int l, int u, int v) // Recursive function implementing binary search

    {

        int mid = (l + u)/2;

        if(u < l) // condition if the search is unsuccessful

        {

            return -1;

        }

        if(v==Adno[mid]) // condition if the search is successful

        {

            return 1;

        }

        else if(v>Adno[mid])

        {

            return binSearch(mid+1,u,v);

        }

        else

        {

            return binSearch(l,mid-1,v);

        }

    }

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

    {

        Admission ob = new Admission();

        System.out.println("Enter Admission number in ascending order");

        ob.fillArray();

        System.out.print("Enter an Admission number to search : ");

        int v = sc.nextInt();

        int f = ob.binSearch(0,99,v);

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

        if(f == 1)

        {

            System.out.println("Admission Number found");

        }

        else

        {

            System.out.println("Admission Number Not found");

        }

    }

}

#### Output:

**Note:** The output has been taken for 5 elements in the array

Enter Admission number in ascending order  
Enter Admission no of student 1: 205  
Enter Admission no of student 2: 310  
Enter Admission no of student 3: 670  
Enter Admission no of student 4: 887  
Enter Admission no of student 5: 952  
Enter an Admission number to search : 887  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
Admission Number found

**Stack operation**

class Stack

{

    int ST[]; // Array to implement stack

    int size; // Maximum size of the stack

    int top; // Index of topmost element (Stack Pointer)

    Stack() // Default constructor

    {

        size = 0;

        top = 0;

    }

    Stack(int cap) // Parameterised Constructor

    {

        size = cap;

        ST = new int[size];

        top = -1; // Initialising top with -1

    }

    void push(int n) // Function to insert element in Stack

    {

        if(top == size-1) // Condition for Overflow

        {

            System.out.println("OVERFLOW");

        }

        else

        {

            top = top + 1;

            ST[top] = n; // Storing value in Stack

        }

    }

    int pop() // Function to delete element from Stack

    {

        if(top == -1) // Condition for Underflow

        {

            System.out.println("UNDERFLOW");

            return -999;

        }

        else

        {

            int val = ST[top]; // Storing the element which will be removed

            top = top - 1;

            return val;

        }

    }

    void display()

    {

        if(top == -1)

        {

            System.out.println("The stack is empty");

        }

        else

        {

            System.out.println("The elements in the stack are : ");

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

            {

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

            }

        }

    }

}

#### Question12:

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 the sentence and reduce all the extra blank space between two words to  
a single blank space.  
(b)        Accept a word from the user which is part of the sentence along with its  
position number and delete the word and display the sentence.

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

**Example 1**

**INPUT**:          A    MORNING WALK IS A IS 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.\*;

class RemoveWord\_ISC2014

{

    public static void main (String args[])

    {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter a sentence : ");

        String s = sc.nextLine();

        s = s.toUpperCase();

        int l = s.length();

        char last = s.charAt(l-1); // Extracting the last character

        /\* Checking whether the sentence ends with '.' or '?' or not \*/

        if(last != '.' && last != '?' && last != '!')

        {

            System.out.println("Invalid Input. End a sentence with either '.', '?' or '!' only");

        }

        else

        {

            StringTokenizer str = new StringTokenizer(s," .?!");

            int c = str.countTokens();

            String w="",ans = "";

            System.out.print("Enter the word to delete : ");

            String del = sc.next();

            System.out.print("Enter the word position is the sentence : ");

            int x = sc.nextInt();

            if(x<1 || x>c) // Checking whether integer inputted is acceptable or not

            {

                System.out.println("Sorry! The word position entered is out of range");

            }

            else

            {

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

                {

                    w = str.nextToken();

                    /\* Skipping if the word to delete and the position matches \*/

                    if(w.equals(del)==true && i == x)

                        continue;

                    ans = ans + w + " ";

                }

                System.out.print("Output : "+ans.trim()+last);

            }

        }

    }

}

#### Output:

**1.** Enter any sentence : A MORNING WALK IS A IS BLESSING FOR THE WHOLE DAY.

Enter the word to delete : IS  
Enter the word position is the sentence : 6

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

**2.** Enter any sentence : STUDY WELL ##  
OUTPUT : Invalid Input. End a sentence with either ‘.’, ‘?’ or ‘!’

#### Question13:

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) Rotate the matrix 90° clockwise

import java.util.\*;

class Q2\_ISC2015

{

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

    {

        Scanner sc=new Scanner(System.in);

        System.out.print("Enter the size of the matrix : ");

        int m=sc.nextInt();

        if(m<3 || m>9)

            System.out.println("Size Out Of Range");

        else

        {

            int A[][]=new int[m][m];

            /\* Inputting the matrix \*/

            for(int i=0;i<m;i++)

            {

                for(int j=0;j<m;j++)

                {

                    System.out.print("Enter an element : ");

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

                }

            }

            /\* Printing the original matrix \*/

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

            System.out.println("The Original Matrix is : ");

            for(int i=0;i<m;i++)

            {

                for(int j=0;j<m;j++)

                {

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

                }

                System.out.println();

            }

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

            /\*Rotation of matrix begins here \*/

            System.out.println("Matrix After Rotation is : ");

            for(int i=0;i<m;i++)

            {

                for(int j=m-1;j>=0;j--)

                {

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

                }

                System.out.println();

            }

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

            int sum = A[0][0]+A[0][m-1]+A[m-1][0]+A[m-1][m-1]; // Finding sum of corner elements

            System.out.println("Sum of the corner elements = "+sum);

        }

    }

}

#### Question14:

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 the sample data and some random data:

**Example 1**

**INPUT :**M = 100  
N = 11

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

import java.util.\*;

class Q1\_ISC2015

{

    int sumDig(long n) // Function to find sum of digits of a number

    {

        int sum = 0, d;

        while(n>0)

        {

            d = (int)(n%10);

            sum = sum + d;

            n = n/10;

        }

        return sum;

    }

    int countDig(long n) // Function to count the number of digits in a number

    {

        String s = Long.toString(n);

        int len = s.length();

        return len;

    }

    public static void main()throws Exception

    {

        Q1\_ISC2015 ob = new Q1\_ISC2015();

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter a value of 'm' from 100 to 10000 : ");

        int m = sc.nextInt();

        System.out.print("Enter a value of n from 1 to 99 : ");

        int n = sc.nextInt();

        if(m<100 || m>10000 || n<1 || n>99)

        {

            System.out.println("Invalid Input");

        }

        else

        {

            long i = (long)m; // Required number can be out of range of 'int'

            /\* The required number must be greater than 'm',

               so loop will go on as long as that number is not obtained.\*/

            while(ob.sumDig(i)!=n)

            {

                i=i+1;

            }

            System.out.println("The required number = "+i);

            System.out.println("Total number of digits = "+ob.countDig(i));

        }

    }

}

#### Output:

Enter a value of ‘m’ from 100 to 10000 : 1500  
Enter a value of ‘n’ from 1 to 99 : 25  
The required number = 1699  
Total number of digits = 4

Enter a value of ‘m’ from 100 to 10000 : 100  
Enter a value of ‘n’ from 1 to 99 : 20  
The required number = 299  
Total number of digits = 3

Enter a value of ‘m’ from 100 to 10000 : 112  
Enter a value of ‘n’ from 1 to 99 : 130  
Invalid Input

#### ****Question15:****

Write a Program in Java to input a 2-D square matrix and check whether it is a Lower Triangular Matrix or not.

**Lower Triangular Matrix :** A Lower Triangular matrix is a square matrix in which all the entries above the main diagonal () are zero. The entries below or on the main diagonal themselves may or may not be zero.

import java.util.\*;

class LowerTriangularMatrix

{

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

    {

        Scanner sc=new Scanner(System.in);

        System.out.print("Enter the size of the matrix : ");

        int m=sc.nextInt();

        int A[][]=new int[m][m];

        /\* Inputting the matrix \*/

        for(int i=0;i<m;i++)

        {

            for(int j=0;j<m;j++)

            {

                System.out.print("Enter an element : ");

                A[i][j]=sc.nextInt();

            }

        }

        /\* Printing the matrix \*/

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

        System.out.println("The Matrix is : ");

        for(int i=0;i<m;i++)

        {

            for(int j=0;j<m;j++)

            {

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

            }

            System.out.println();

        }

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

        int p=0;

        for(int i=0;i<m;i++)

        {

            for(int j=i+1;j<m;j++)

            {

                /\* Checking that the matrix is Lower Triangular or not \*/

                if(A[i][j]!=0) // All elements above the diagonal must be zero

                {

                    p=1;

                    break;

                }

            }

        }

        if(p==0)

            System.out.println("The matrix is Lower Triangular");

        else

            System.out.println("The matrix is not Lower Triangular");

    }

}

#### ****Output:****

Enter the size of the matrix : 4  
Enter an element : 5  
Enter an element : 0  
Enter an element : 0  
Enter an element : 0  
Enter an element : 3  
Enter an element : 1  
Enter an element : 0  
Enter an element : 0  
Enter an element : 4  
Enter an element : 9  
Enter an element : 4  
Enter an element : 0  
Enter an element : 6  
Enter an element : 8  
Enter an element : 7  
Enter an element : 2  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
The Matrix is :  
5 0 0 0  
3 1 0 0  
4 9 4 0  
6 8 7 2  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
The matrix is Lower Triangular

#### ****Question16:****

Write a Program in Java to input a 2-D square matrix and check whether it is an Upper Triangular Matrix or not.

**Upper Triangular Matrix :** An Upper Triangular matrix is a square matrix in which all the entries below the main diagonal () are zero. The entries above or on the main diagonal themselves may or may not be zero.

import java.util.\*;

class UpperTriangularMatrix

{

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

    {

        Scanner sc=new Scanner(System.in);

        System.out.print("Enter the size of the matrix : ");

        int m=sc.nextInt();

        int A[][]=new int[m][m];

        /\* Inputting the matrix \*/

        for(int i=0;i<m;i++)

        {

            for(int j=0;j<m;j++)

            {

                System.out.print("Enter an element : ");

                A[i][j]=sc.nextInt();

            }

        }

        /\* Printing the matrix \*/

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

        System.out.println("The Matrix is : ");

        for(int i=0;i<m;i++)

        {

            for(int j=0;j<m;j++)

            {

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

            }

            System.out.println();

        }

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

        int p=0;

        for(int i=0;i<m;i++)

        {

            for(int j=0;j<i;j++)

            {

                /\* Checking that the matrix is Upper Triangular or not \*/

                if(A[i][j]!=0) // All elements below the diagonal must be zero

                {

                    p=1;

                    break;

                }

            }

        }

        if(p==0)

            System.out.println("The matrix is Upper Triangular");

        else

            System.out.println("The matrix is not Upper Triangular");

    }

}

#### ****Output:****

Enter the size of the matrix : 4  
Enter an element : 5  
Enter an element : 3  
Enter an element : 0  
Enter an element : 7  
Enter an element : 0  
Enter an element : 1  
Enter an element : 9  
Enter an element : 8  
Enter an element : 0  
Enter an element : 0  
Enter an element : 4  
Enter an element : 6  
Enter an element : 0  
Enter an element : 0  
Enter an element : 0  
Enter an element : 2  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
The Matrix is :  
5 3 0 7  
0 1 9 8  
0 0 4 6  
0 0 0 2  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
The matrix is Upper Triangular

#### ****Question17:****

Write a Program in Java to input a number and check whether it is a Bouncy Number or not.

**Increasing Number :** Working from left-to-right if no digit is exceeded by the digit to its left it is called an increasing number; for example, 22344.  
**Decreasing Number :** Similarly if no digit is exceeded by the digit to its right it is called a decreasing number; for example, 774410.  
**Bouncy Number :** We shall call a positive integer that is neither increasing nor decreasing a “bouncy” number; for example, 155349. Clearly there cannot be any bouncy numbers below 100.

import java.util.\*;

class BouncyNumber

{

    boolean isIncreasing(int n) //Function to check whether a number is Increasing

    {

        String s = Integer.toString(n);

        char ch;

        int f = 0;

        for(int i=0; i<s.length()-1; i++)

        {

            ch = s.charAt(i);

            if(ch>s.charAt(i+1))// If any digit is more than next digit then we have to stop checking

            {

                f = 1;

                break;

            }

        }

        if(f==1)

            return false;

        else

            return true;

    }

    boolean isDecreasing(int n) //Function to check whether a number is Decreasing

    {

        String s = Integer.toString(n);

        char ch;

        int f = 0;

        for(int i=0; i<s.length()-1; i++)

        {

            ch = s.charAt(i);

            if(ch<s.charAt(i+1))// If any digit is less than next digit then we have to stop checking

            {

                f = 1;

                break;

            }

        }

        if(f==1)

            return false;

        else

            return true;

    }

    void isBouncy(int n)

    {

        if(isIncreasing(n)==true)

            System.out.println("The number " + n + " is Increasing and Not Bouncy");

        else if(isDecreasing(n)==true)

            System.out.println("The number " + n + " is Decreasing and Not Bouncy");

        else

            System.out.println("The number " + n + " is bouncy");

    }

    public static void main(String args[])

    {

        Scanner sc = new Scanner(System.in);

        BouncyNumber ob = new BouncyNumber();

        System.out.print("Enter a number : ");

        int n = sc.nextInt();

        ob.isBouncy(n);

    }

}

#### ****Output:****

Enter a number : 22344  
The number 22344 is Increasing and Not Bouncy  
Enter a number : 774410  
The number 774410 is Decreasing and Not Bouncy  
Enter a number : 155349  
The number 155349 is bouncy

#### ****Question18:****

Write a Program in Java to input a 2-D square matrix and check whether it is a Diagonal Matrix or not.

**Diagonal Matrix :** A diagonal matrix is a matrix (usually a square matrix) in which the entries outside the main diagonal () are all zero. The diagonal entries themselves may or may not be zero (but all diagonal entries cannot be zero).

import java.util.\*;

class DiagonalMatrix

{

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

{

Scanner sc=new Scanner(System.in);

System.out.print("Enter the size of the matrix : ");

int m=sc.nextInt();

int A[][]=new int[m][m];

/\* Inputting the matrix \*/

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

{

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

{

System.out.print("Enter an element : ");

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

}

}

/\* Printing the matrix \*/

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

System.out.println("The Matrix is : ");

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

{

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

{

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

}

System.out.println();

}

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

int p=0, q=0;

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

{

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

{

if(i!=j && A[i][j]!=0) // Checking non-diagonal elements

{

p=1;

break;

}

if(i==j && A[i][j]==0) // Checking diagonal elements

{

q++;

}

}

}

if(p==0 && q<m)

System.out.println("The matrix is Diagonal");

else

System.out.println("The matrix is not Diagonal");

}

}

#### ****Output:****

Enter the size of the matrix : 4  
Enter an element : 5  
Enter an element : 0  
Enter an element : 0  
Enter an element : 0  
Enter an element : 0  
Enter an element : 1  
Enter an element : 0  
Enter an element : 0  
Enter an element : 0  
Enter an element : 0  
Enter an element : 0  
Enter an element : 0  
Enter an element : 0  
Enter an element : 0  
Enter an element : 0  
Enter an element : 7  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
The Matrix is :  
5 0 0 0  
0 1 0 0  
0 0 0 0  
0 0 0 7  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
The matrix is Diagonal

#### ****Question19:****

Write a Program in Java to input two 2-D arrays and perform Matrix Multiplication:

import java.util.\*;

class MatrixMultiplication

{

    void printMatrix(int P[][], int r, int c) // Funtion for printing an array

    {

        for(int i=0; i<r; i++)

        {

            for(int j=0; j<c; j++)

            {

                System.out.print(P[i][j]+"\t");

            }

            System.out.println();

        }

    }

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

    {

        MatrixMultiplication ob = new MatrixMultiplication();

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter no. of rows of 1st Matrix : ");

        int r1=sc.nextInt();

        System.out.print("Enter no. of columns of 1st Matrix : ");

        int c1=sc.nextInt();

        System.out.print("Enter no. of rows of 2nd Matrix : ");

        int r2=sc.nextInt();

        System.out.print("Enter no. of columns of 2nd Matrix : ");

        int c2=sc.nextInt();

        if(c1 != r2) // Condition for Multiplication to be possible

        {

            System.out.println("Matrix Multiplication of the given order is not possible");

        }

        else

        {

            int A[][]=new int[r1][c1]; // Array to store 1st Matrix

            int B[][]=new int[r2][c2]; // Array to store 2nd Matrix

            int C[][]=new int[r1][c2]; // Array to store Result of Multiplication of 2 Matrices

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

            System.out.println("Inputting the 1st Matrix");

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

            for(int i=0; i<r1; i++)

            {

                for(int j=0; j<c1; j++)

                {

                    System.out.print("Enter an element : ");

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

                }

            }

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

            System.out.println("Inputting the 2nd Matrix");

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

            for(int i=0; i<r2; i++)

            {

                for(int j=0; j<c2; j++)

                {

                    System.out.print("Enter an element : ");

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

                }

            }

            /\* Matrix Multiplication Starts Here \*/

            int sum = 0;

            for(int i=0; i<r1; i++)

            {

                for(int j=0; j<c2; j++)

                {

                    for(int k=0; k<c1; k++)

                    {

                        sum = sum + A[i][k]\*B[k][j];

                    }

                    C[i][j]=sum;

                    sum=0;

                }

            }

            /\* Printing all the Matrices \*/

            System.out.println("n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

            System.out.println("          Output         ");

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

            System.out.println("The 1st Matrix is");

            ob.printMatrix(A,r1,c1);

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

            System.out.println("The 2nd Matrix is");

            ob.printMatrix(B,r2,c2);

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

            System.out.println("The Result of Multiplication is");

            ob.printMatrix(C,r1,c2);

        }

    }

}

#### ****Output:****

Enter no. of rows of 1st Matrix : 2  
Enter no. of columns of 1st Matrix : 3  
Enter no. of rows of 2nd Matrix : 3  
Enter no. of columns of 2nd Matrix : 2  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
Inputting the 1st Matrix  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
Enter an element : 8  
Enter an element : 1  
Enter an element : 2  
Enter an element : -5  
Enter an element : 6  
Enter an element : 7  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
Inputting the 2nd Matrix  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
Enter an element : -5  
Enter an element : 1  
Enter an element : 0  
Enter an element : 2  
Enter an element : -11  
Enter an element : 7

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
Output  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
The 1st Matrix is  
8 1 2  
-5 6 7  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
The 2nd Matrix is  
-5 1  
0 2  
-11 7  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
The Result of Multiplication is  
-62 24  
-52 56

#### Question20:

Accept a paragraph of text consisting of sentences that are terminated by either ‘.’ (full stop), ‘!’ (exclamation mark) or a ‘?’ (question mark). Assume that there can be maximum 10 sentences in a paragraph. Write a program to arrange the sentences in increasing order of their number of words.

**Example :**  
**INPUT :**Please come and attend the party. Hello! How are you?  
**OUTPUT :**  
Hello = 1  
How are you = 3  
Please come and attend the party = 6

import java.util.\*;

class sortParagraph

{

    // Function to count no. of words in every sentence

    int countWords(String s)

    {

        StringTokenizer str = new StringTokenizer(s," .,?!");

        int c = str.countTokens();

        return c;

    }

    // Function to sort the sentences in ascending order of their no. of words

    void sort(String w[], int p[])

    {

        int n = w.length, t1 = 0;

        String t2 = "";

        for(int i=0; i<n-1; i++)

        {

            for(int j=i+1; j<n; j++)

            {

                if(p[i]>p[j]) // for descending use p[i]<p[j]

                {

                    t1 = p[i];

                    p[i] = p[j];

                    p[j] = t1;

                    t2 = w[i];

                    w[i] = w[j];

                    w[j] = t2;

                }

            }

        }

        printResult(w,p);    // Calling function for printing the result

    }

    void printResult(String w[], int p[]) // Function to print the final result

    {

        int n = w.length;

        for(int i=0; i<n; i++)

        {

            System.out.println(w[i]+"\t=\t"+p[i]);

        }

    }

    public static void main(String args[])

    {

        sortParagraph ob = new sortParagraph();

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter a paragraph : "); //Inputting a paragraph

        String pg = sc.nextLine();

        StringTokenizer str = new StringTokenizer(pg,".?!");

        int count = str.countTokens(); //Counting no. of sentences in the paragraph

        if(count > 10)

            System.out.println("A maximum of 10 sentences are allowed in the paragraph");

        else

        {

            String sent[] = new String[count]; //Array to store the sentences separately

            int p[] = new int[count]; //Array to store no. of words of each sentence

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

            {

                sent[i] = str.nextToken().trim(); // Saving sentences one by one in an array

                p[i] = ob.countWords(sent[i]); // Saving no. of words of every sentence

            }

            ob.sort(sent,p);

        }

    }

}

#### ****Question21:****

Write a program to declare a square matrix A[ ][ ] of order ‘n’. Allow the user to input positive integers into this matrix. Perform the following tasks on the matrix:

(i) Output the original matrix.  
(ii) Find the SADDLE POINT for the matrix. If the matrix has no saddle point, output the message “NO SADDLE POINT”.

[**Note:** A **saddle point** is an element of the matrix such that it is the **minimum element for the row** to which it belongs and the **maximum element for the column** to which it belongs. Saddle point for a given matrix is always unique.]

**Example:** In the Matrix  
4 5 6  
7 8 9  
5 1 3

Saddle point = 7 because it is the minimum element of row 2 and maximum element of column 1

import java.io.\*;

class SaddlePoint

{

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

   {

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

       System.out.print("Enter the order of the matrix : ");

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

       int A[][]=new int[n][n];

       System.out.println("Inputting the elements in the matrix");

       System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"); // Ignore these. They are just for styling

       for(int i=0;i<n;i++)

       {

         for(int j=0;j<n;j++)

         {

             System.out.print("Enter Element at ["+i+"]["+j+"] : ");

             A[i][j]=Integer.parseInt(br.readLine());

         }

       }

       /\* Printing the Original Matrix \*/

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

       System.out.println("The Original Matrix is");

       for(int i=0;i<n;i++)

       {

         for(int j=0;j<n;j++)

         {

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

         }

         System.out.println();

       }

       int max, min, x, f=0;

       for(int i=0;i<n;i++)

       {

           /\* Finding the minimum element of a row \*/

           min = A[i][0]; // Initializing min with first element of every row

           x = 0;

           for(int j=0;j<n;j++)

           {

                if(A[i][j]<min)

                {

                    min = A[i][j];

                    x = j; // Saving the column position of the minimum element of the row

                }

           }

           /\* Finding the maximum element in the column

            \* corresponding to the minimum element of row \*/

           max = A[0][x]; // Initializing max with first element of that column

           for(int k=0;k<n;k++)

           {

                if(A[k][x]>max)

                {

                    max = A[k][x];

                }

           }

            /\* If the minimum of a row is same as maximum of the corresponding column,

               then, we have that element as the Saddle point \*/

           if(max==min)

           {

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

               System.out.println("Saddle point = "+max);

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

               f=1;

           }

       }

       if(f==0)

       {

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

           System.out.println("No saddle point");

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

       }

   }

}

#### ****Question22:****

Write a Program in Java to print all the **Twin Prime** numbers within a given range.

**Note:** Twin Prime numbers are a pair of numbers which are both prime and their difference is 2.

**Example:** Twin Prime numbers in the range 1 to 100 are :  
(3,5) (5,7) (11,13) (17,19) (29,31) (41,43) (59,61) (71,73)

import java.io.\*;

class TwinPrimeRange

{

     boolean isPrime(int n) //funton for checking prime

        {

            int count=0;

            for(int i=1; i<=n; i++)

                {

                    if(n%i == 0)

                        count++;

                }

            if(count == 2)

                return true;

             else

                return false;

        }

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

        {

            TwinPrimeRange ob = new TwinPrimeRange();

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

            System.out.print("Enter the lower range : ");

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

            System.out.print("Enter the upper range : ");

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

            if(p>q)

                System.out.println("Invalid Range !");

            else

            {

                System.out.println("nThe Twin Prime Numbers within the given range are : ");

                for(int i=p; i<=(q-2); i++)

                {

                    if(ob.isPrime(i) == true && ob.isPrime(i+2) == true)

                    {

                        System.out.print("("+i+","+(i+2)+") ");

                    }

                }

            }

        }

#### }

#### ****Output:****

Enter the lower range : 1  
Enter the upper range : 200

The Twin Prime Numbers within the given range are :  
(3,5) (5,7) (11,13) (17,19) (29,31) (41,43) (59,61) (71,73) (101,103) (107,109) (137,139) (149,151) (179,181) (191,193) (197,199)

#### Question23:

A sequence of Fibonacci Strings is generated as follows:

S0 = “a”, S1 = “b”, Sn = S(n-1) + S(n-2) where ‘+’ denotes concatenation. Thus the sequence is:

a, b, ba, bab, babba, babbabab, ………. n terms.

Design a class **FiboString** to generate Fibonacci strings. Some of the members of the class are given below:

**Class name         :**              FiboString

**Data members/instance variables:**

x                              :               to store the first string  
y                              :               to store the second string  
z                              :               to store the concatenation of the previous two strings  
n                             :               to store the number of terms

**Member functions/methods:**

FiboString()        :               constructor to assign x=”a”, y=”b”, z=”ba”  
void accept()      :               to accept the number of terms ‘n’  
void generate()   :               to generate and print the fibonacci strings. The sum of (‘+’ i.e. concatenation) first two strings is the third string. Eg. “a” is first string, “b” is second string then the third string will be “ba” and fourth will be “bab” and so on.

Specify the class **FiboString**, giving details of the **constructor()**, **void accept()** and **void generate()**. Define the **main()** function to create an object and call the functions accordingly to enable the task.

import java.io.\*;

class FiboString

{

    String x,y,z;

    int n;

    FiboString() // Constructor

    {

        x = "a";

        y = "b";

        z = "ba"; // mentioned in the question otherwise not required. z = "" is sufficient

    }

    void accept()throws IOException

    {

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

        System.out.print("\nEnter the number of terms : ");

        n = Integer.parseInt(br.readLine());

    }

    void generate()

    {

        System.out.print("nThe Fibonacci String Series is : ");

        if(n <= 1) // If no of terms is less than or equal to 1

            System.out.print(x);

        else // If no of terms is more than or equal to 2

        {

            System.out.print(x+", "+y);

            for(int i=3; i<=n; i++)

            {

                z = y+x;

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

                x = y;

                y = z;

            }

        }

    }

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

    {

        FiboString ob = new FiboString();

        ob.accept();

        ob.generate();

    }

#### }

#### Output:

**1)**Enter the number of terms : 2  
The Fibonacci String Series is : a, b

**2)** Enter the number of terms : 5  
The Fibonacci String Series is : a, b, ba, bab, babba

#### Question24:

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 that has more than two 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 two 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

import java.io.\*;

class MagicComposite\_ISC2014

{

    boolean isComposite(int n) // Function to check for Composite number

    {

        int count=0;

        for(int i=1;i<=n;i++)

        {

            if(n%i==0)

                count++;

        }

        if(count>2)

            return true;

        else

            return false;

    }

    int sumDig(int n) // Function to return sum of digits of a number

    {

        int s = 0;

        while(n>0)

        {

            s = s + n%10;

            n = n/10;

        }

        return s;

    }

    boolean isMagic(int n) // Function to check for Magic number

    {

        int a = sumDig(n);

        while(a>9)

        {

            a = sumDig(a);

        }

        if(a == 1)

            return true;

        else

            return false;

    }

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

    {

        MagicComposite\_ISC2014 ob = new MagicComposite\_ISC2014();

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

        System.out.print("Enter the lower limit(m) : ");

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

        System.out.print("Enter the upper limit(n) : ");

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

        int c=0;

        if (m<n)

        {

            System.out.println("The Composite Magic Integers are: ");

            for(int i=m; i<=n; i++)

            {

                if(ob.isComposite(i)==true && ob.isMagic(i)==true)

                {

                    if (c==0) // Printing the first number without any comma

                        System.out.print(i);

                    else

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

                    c++;

                }

            }

            System.out.println("nThe frequency of Composite Magic Integers is : "+c);

        }

        else

            System.out.println("OUT OF RANGE");

    }

}

#### Output:

Enter the lower limit(m) : 1200  
Enter the upper limit(n) : 1300  
The Composite Magic Integers are:  
1207, 1216, 1225, 1234, 1243, 1252, 1261, 1270, 1288  
The frequency of Composite Magic Integers is : 9

#### Question26:

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 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 of the ith row and jth column is equal to the element of the jth row and ith 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 with the sample data and some random data:

**Example 1**

**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

import java.io.\*;

class SymetricMatrix\_ISC2014

{

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

   {

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

       System.out.print("Enter the number of elements : ");

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

       int A[][]=new int[m][m];

       if(m>2 && m<10) // Checking for valid input of rows and columns size

       {

           System.out.println("\nInputting the elements in the Matrix: n");

           for(int i=0;i<m;i++)

           {

               for(int j=0;j<m;j++)

               {

                   System.out.print("Enter the elements : ");

                   A[i][j]=Integer.parseInt(br.readLine());

               }

           }

           /\* Printing the Original Matrix \*/

           System.out.println("\nThe Original Matrix is : ");

           for(int i=0;i<m;i++)

           {

               for(int j=0;j<m;j++)

               {

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

               }

               System.out.println();

           }

           /\* Checking whether the matrix is symmetric or not \*/

           int flag = 0;

           for(int i=0;i<m;i++)

           {

               for(int j=0;j<m;j++)

               {

                   if(A[i][j] != A[j][i])

                   {

                       flag = 1; // Setting flag = 1 when elements do not match

                       break;

                    }

               }

           }

           if(flag == 1)

                System.out.println("\nThe given Matrix is Not Symmetric");

           else

                System.out.println("\nThe given Matrix is Symmetric");

                /\* Finding sum of the diagonals \*/

                int ld = 0, rd = 0;

                for(int i=0;i<m;i++)

                {

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

                    {

                        if(i == j) // Condition for the left diagonal

                        {

                            ld = ld + A[i][j];

                        }

                        if((i+j) == (m-1)) // Condition for the right diagonal

                        {

                            rd = rd + A[i][j];

                        }

                    }

                }

                System.out.println("The sum of the left diagonal = "+ld);

                System.out.println("The sum of the right diagonal = "+rd);

       }

       else

            System.out.println("The Matrix Size is Out Of Range");

   }

}

#### Question27:

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 the sentence and reduce all the extra blank space between two words to  
a single blank space.  
(b)        Accept a word from the user which is part of the sentence along with its  
position number and delete the word and display the sentence.

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

**Example 1**

**INPUT**:          A    MORNING WALK IS A IS 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.\*;  class RemoveWord\_ISC2014  {      public static void main (String args[])      {          Scanner sc = new Scanner(System.in);            System.out.print("Enter a sentence : ");          String s = sc.nextLine();          s = s.toUpperCase();          int l = s.length();          char last = s.charAt(l-1); // Extracting the last character            /\* Checking whether the sentence ends with '.' or '?' or not \*/          if(last != '.' && last != '?' && last != '!')          {              System.out.println("Invalid Input. End a sentence with either '.', '?' or '!' only");          }          else          {              StringTokenizer str = new StringTokenizer(s," .?!");              int c = str.countTokens();              String w="",ans = "";              System.out.print("Enter the word to delete : ");              String del = sc.next();              System.out.print("Enter the word position is the sentence : ");              int x = sc.nextInt();                if(x<1 || x>c) // Checking whether integer inputted is acceptable or not              {                  System.out.println("Sorry! The word position entered is out of range");              }              else              {                  for(int i=1; i<=c; i++)                  {                      w = str.nextToken();                      /\* Skipping if the word to delete and the position matches \*/                      if(w.equals(del)==true && i == x)                          continue;                      ans = ans + w + " ";                  }                  System.out.print("Output : "+ans.trim()+last);              }          }      }  } |

Output:

**1.** Enter any sentence : A MORNING WALK IS A IS BLESSING FOR THE WHOLE DAY.

Enter the word to delete : IS  
Enter the word position is the sentence : 6

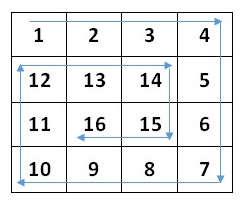
Output : A MORNING WALK IS A BLESSING FOR THE WHOLE DAY.

**2.** Enter any sentence : STUDY WELL ##  
OUTPUT : Invalid Input. End a sentence with either ‘.’, ‘?’ or ‘!’

#### ****Question28:****

Write a Program in Java to fill a square matrix of size ‘n\*n” in a circular fashion (clockwise) with natural numbers from 1 to n\*n, taking ‘n’ as input.

For example: if n = 4, then n\*n = 16, hence the array will be filled as given below.

[](http://www.guideforschool.com/wp-content/uploads/2014/02/spiral-circular-matrix.jpg)

import java.util.\*;

class Circular\_Matrix

{

public static void main(String args[])

{

Scanner sc = new Scanner(System.in);

System.out.print("Enter the number of elements : ");

int n = sc.nextInt();

int A[][] = new int[n][n];

int k=1, c1=0, c2=n-1, r1=0, r2=n-1;

while(k<=n\*n)

{

for(int i=c1;i<=c2;i++)

{

A[r1][i]=k++;

}

for(int j=r1+1;j<=r2;j++)

{

A[j][c2]=k++;

}

for(int i=c2-1;i>=c1;i--)

{

A[r2][i]=k++;

}

for(int j=r2-1;j>=r1+1;j--)

{

A[j][c1]=k++;

}

c1++;

c2--;

r1++;

r2--;

}

/\* Printing the Circular matrix \*/

System.out.println("The Circular Matrix is:");

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

{

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

{

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

}

System.out.println();

}

}

}

#### ****Question29:****

Write a Program in Java to fill a 2-D array with the first ‘m\*n’ prime numbers, where ‘m’ is the number of rows and ‘n’ is the number of columns.

For example: If rows = 4 and columns = 5, then the result should be:

/\*\*

\* The class FillPrime fills a 2D array with 'm\*n' Prime numbers

\* @author : www.javaforschool.com

\* @Program Type : BlueJ Program - Java

\*/

import java.io.\*;

class FillPrime

{

boolean isPrime(int n) // Function for checking whether a number is prime or not

{

int c = 0;

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

{

if(n%i == 0)

c++;

}

if(c == 2)

return true;

else

return false;

}

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

{

FillPrime ob = new FillPrime();

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

System.out.print("Enter the number of rows: ");

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

System.out.print("Enter the number of columns: ");

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

int A[][]=new int[m][n]; // 2D array for storing 'm\*n' prime numbers

int B[] = new int [m\*n]; // 1D array for storing 'm\*n' prime numbers

int i = 0, j;

int k = 1; // For generating natural numbers

/\* First saving the 'm\*n' prime numbers into a 1D Array \*/

while(i < m\*n)

{

if(ob.isPrime(k)==true)

{

B[i] = k;

i++;

}

k++;

}

/\* Saving the 'm\*n' prime numbers from 1D array into the 2D Array \*/

int x = 0;

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

{

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

{

A[i][j] = B[x];

x++;

}

}

/\* Printing the resultant 2D array \*/

System.out.println("The Filled Array is :");

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

{

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

{

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

}

System.out.println();

}

}

}

#### Question30:

Write a program in Java to input the first 14 digits of an IMEI number and find the check (last) digit of it.

import java.io.\*;

class IMEI\_CheckDig

{

    int sumDig(int n) // Function for finding and returning sum of digits of a number

    {

        int a = 0;

        while(n>0)

        {

            a = a + n%10;

            n = n/10;

        }

        return a;

    }

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

    {

        IMEI\_CheckDig ob = new IMEI\_CheckDig();

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

        System.out.print("Enter first 14 digits of IMEI code : ");

        long n = Long.parseLong(br.readLine()); // 14 digits cannot be stored in 'int' data type

        String s = Long.toString(n); // Converting the number into String for finding length

        int l = s.length();

        if(l!=14) // If length is not 14 then IMEI is Invalid

            System.out.println("Output : Invalid Input");

        else

        {

            int d = 0, sum = 0;

            for(int i=14; i>=1; i--)

            {

                d = (int)(n%10);

                if(i%2 == 0)

                {

                    d = 2\*d; // Doubling every alternate digit

                }

                sum = sum + ob.sumDig(d); // Finding sum of the digits

                n = n/10;

            }

            System.out.println("Sum = "+sum);

            int dig = (9\*sum)%10; // Finding the check digit

            System.out.println("Output : The check digit = "+dig);

        }

    }

}

#### Output:

Enter first 14 digits of IMEI code : 49015420323751  
Sum = 52  
Output : The check digit = 8

#### Question31:

The **International Mobile Station Equipment Identity** or **IMEI** is a number, usually unique, to identify mobile phones, as well as some satellite phones. It is usually found printed inside the battery compartment of the phone.

The IMEI number is used by a GSM network to identify valid devices and therefore can be used for stopping a stolen phone from accessing that network.

The IMEI (15 decimal digits: 14 digits plus a check digit) includes information on the origin, model, and serial number of the device.

**See:** Java Program to find check digit of an IMEI Number

The IMEI is validated in three steps:

Starting from the right, double every other digit (e.g., 7 becomes 14).

Sum the digits (e.g., 14 → 1 + 4).

Check if the sum is divisible by 10.

import java.io.\*;

class IMEI

{

    int sumDig(int n) // Function for finding and returning sum of digits of a number

    {

        int a = 0;

        while(n>0)

        {

            a = a + n%10;

            n = n/10;

        }

        return a;

    }

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

    {

        IMEI ob = new IMEI();

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

        System.out.print("Enter a 15 digit IMEI code : ");

        long n = Long.parseLong(br.readLine()); // 15 digits cannot bestored in 'int' data type

        String s = Long.toString(n); // Converting the number intoString for finding length

        int l = s.length();

        if(l!=15) // If length is not 15 then IMEI is Invalid

            System.out.println("Output : Invalid Input");

        else

        {

            int d = 0, sum = 0;

            for(int i=15; i>=1; i--)

            {

                d = (int)(n%10);

                if(i%2 == 0)

                {

                    d = 2\*d; // Doubling every alternate digit

                }

                sum = sum + ob.sumDig(d); // Finding sum of the digits

                n = n/10;

            }

            System.out.println("Output : Sum = "+sum);

            if(sum%10==0)

                System.out.println("Valid IMEI Code");

            else

System.out.println("Invalid IMEI Code");

 }

 }

}

#### Output:

**1.** Enter a 15 digit IMEI code : 654122487458946  
Output : Sum = 80  
Valid IMEI Code

**2.**Enter a 15 digit IMEI code : 799273987135461  
Output : Sum = 79  
Invalid IMEI Code

#### ****Question32:****

Write a Program in Java to input a 2-D array of size ‘m\*n’ and print its boundary (border) elements.

import java.io.\*;

class Boundary\_Element

{

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

    {

        int i,j,m,n;

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

        System.out.print("Enter the no. of  rows: "); //Inputting the number of rows

        m=Integer.parseInt(br.readLine());

        System.out.print("Enter the no. of columns: "); //Inputting the number of columns

        n=Integer.parseInt(br.readLine());

        int A[][]=new int[m][n]; //Creating the array

        /\* Inputting the array \*/

        for(i=0;i<m;i++)

        {

            for(j=0;j<n;j++)

            {

                System.out.print("Enter the elements: ");

                A[i][j]=Integer.parseInt(br.readLine());

            }

        }

        System.out.println("The Boundary Elements are:");

        for(i=0;i<m;i++)

        {

            for(j=0;j<n;j++)

            {

                if(i==0 || j==0 || i == m-1 || j == n-1) //condition for accessing boundary elements

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

                else

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

            }

            System.out.println();

        }

    }

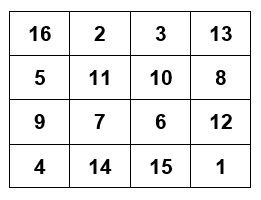
}

#### ****Question33:****

A square matrix is said to be a Magic Square, if the sum of each row, each column and each diagonal is same. Write a program to enter an integer number ‘n’. Create a magic square of size ‘n\*n’. Finally, print the elements of the matrix as Magic Square.

**Note:** n <= 5

**Sample Input:**Enter the size of the matrix : 4  
**Sample Output:**The Magic Matrix of size 4×4 is:

[](http://www.guideforschool.com/wp-content/uploads/2014/02/magic-matrix-1.jpg)

import java.io.\*;

class Magic\_Matrix

    {

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

        {

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

            System.out.print("\n\nEnter the size of the matrix : ");

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

            if(n>5)

             System.out.println("Enter a number between 1 to 5 ");

            else

            {

            int A[][]=new int[n][n]; // Creating the Magic Matrix

            int i,j,k,t;

            /\*Initializing every cell of the matrix with 0 \*/

            for(i=0;i<n;i++)

                {

                    for(j=0;j<n;j++)

                        {

                            A[i][j] = 0;

                        }

                }

            /\* When the size of the matrix is Odd \*/

            if(n%2!=0)

            {

                i=0;

                j = n/2;

                k = 1;

                while(k<=n\*n)

                {

                    A[i][j] = k++;

                    i--; // Making one step upward

                    j++; // Moving one step to the right

                    if(i<0 && j>n-1) // Condition for the top-right corner element

                    {

                        i = i+2;

                        j--;

                    }

                    if(i<0) // Wrapping around the row if it goes out of boundary

                        i = n-1;

                    if(j>n-1) // Wrapping around the column if it goes out of boundary

                        j = 0;

                    if(A[i][j]>0) // Condition when the cell is already filled

                    {

                        i = i+2;

                        j--;

                    }

                }

            }

            /\* When the size of the matrix is even \*/

            else

            {

                k = 1;

                /\* Filling the matrix with natural numbers from 1 till n\*n \*/

                for(i=0;i<n;i++)

                {

                    for(j=0;j<n;j++)

                        {

                            A[i][j] = k++;

                        }

                }

                j = n-1;

                for(i=0; i<n/2; i++)

                {

                    /\* swapping corner elements of primary diagonal \*/

                    t = A[i][i];

                    A[i][i] = A[j][j];

                    A[j][j] = t;

                    /\* swapping corner elements of secondary diagonal \*/

                    t = A[i][j];

                    A[i][j] = A[j][i];

                    A[j][i] = t;

                    j--;

                }

            }

            /\* Printing the Magic matrix \*/

            System.out.println("The Magic Matrix of size "+n+"x"+n+" is:");

            for(i=0;i<n;i++)

                {

                    for(j=0;j<n;j++)

                        {

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

                        }

                 System.out.println();

                }

         }

        }

    }

#### ****Question34:****

Given a square matrix M [ ] [ ] of order ‘n’. The maximum value possible for ‘n’ is 10. Accept three different characters from the keyboard and fill the array according to the instruction given below.

Fill the upper and lower elements formed by the intersection of the diagonals by character 1.  
Fill the left and right elements formed by the intersection of the diagonals by character 2.  
Fill both the diagonals by character 3.

Output the result in format given below:

**Example 1**

ENTER SIZE : 4  
INPUT : FIRST CHARACTER : ‘\*’  
SECOND CHARACTER : ‘?’  
THIRD CHARACTER : ‘#’

OUTPUT :

|  |
| --- |
| # \* \* #  ? # # ?  ? # # ?  # \* \* # |

**Example 2**

ENTER SIZE : 5  
INPUT : FIRST CHARACTER : ‘$’  
SECOND CHARACTER : ‘!’  
THIRD CHARACTER : ‘@’

OUTPUT :

|  |
| --- |
| @ $ $ $ @  ! @ $ @ !  ! ! @ ! !  ! @ $ @ !  @ $ $ $ @ |

**Example 3**

ENTER SIZE : 65  
OUTPUT : SIZE OUT OF RANGE

import java.util.\*;

class MatrixFill

{

    public static void main(String args[])

    {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter size of the matrix : ");

        int n = sc.nextInt();

        if(n<2 || n>10)

            System.out.println("Size out of Range");

        else

        {

            char A[][]=new char[n][n];

            System.out.print("Enter the 1st character : ");

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

            System.out.print("Enter the 2nd character : ");

            char c2 = sc.next().charAt(0);

            System.out.print("Enter the 3rd character : ");

            char c3 = sc.next().charAt(0);

            for(int i=0; i<n; i++)

            {

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

                {

                    if(i==j || (i+j)==(n-1))

                        A[i][j] = c3; // Filling the diagonals with 3rd character

                    else

                        A[i][j] = c2; // Filling all other positions with 2nd character

                }

            }

            for(int i=0; i<n/2; i++)

            {

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

                {

                    A[i][j] = c1; // Filling the upper positions formed by intersection of diagonals

                    A[n-1-i][j] = c1; // Filling the lower positions formed by intersection of diagonals

                }

            }

            // Printing the Matrix

            System.out.println("\nOutput : \n");

            for(int i=0; i<n; i++)

            {

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

                {

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

                }

                System.out.println();

            }

        }

    }

}

**Output:**

|  |
| --- |
| Enter size of the matrix : 7  Enter the 1st character : @  Enter the 2nd character : #  Enter the 3rd character : %    Output :    % @ @ @ @ @ %  # % @ @ @ % #  # # % @ % # #  # # # % # # #  # # % @ % # #  # % @ @ @ % #  % @ @ @ @ @ % |

#### ****Question35:****

Write a Program in Java to input a Date in ddmmyyyy 8-digit format and print it in:  
1) dd/mm/yyyy format  
2) dd, month name, yyyy format

**Input:** 01011943  
**Output:**  
01/10/1943  
1 January, 1943

import java.io.\*;

class Date\_DDMMYY

{

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

    {

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

        int l, y, d, m;

        String dd, mm, yy;

        //array storing the maximum days of every month

        int maxdays[]={0,31,28,31,30,31,30,31,31,30,31,30,31};

        //array storing the month names

        String month[]={ "", "January", "February", "March", "April", "May", "June", "July", "August",

                         "September", "October", "November", "December" };

        System.out.print("Enter any date in 8 digits (ddmmyyyy) format: ");

        String date = br.readLine(); //inputting the date in String format

        l = date.length(); //finding number of digits in the given input

        if(l==8) //performing the task only when number of digits is 8

        {

            dd = date.substring(0,2); //extracting the day in String format

            mm = date.substring(2,4); //extracting the month in String format

            yy = date.substring(4); //extracting the year in String format

            d = Integer.parseInt(dd); //day in Integer format

            m = Integer.parseInt(mm); //month in Integer format

            y = Integer.parseInt(yy); //year in Integer format

            if((y%400==0) || ((y%100!=0)&&(y%4==0))) // condition for leap year

            {

                maxdays[2]=29;

            }

            /\* checking whether the day, month and year are within acceptable range

               i.e. there cannot be an input like 35012013 because 35/01/2013 is unacceptable\*/

            if(m<0 || m>12 || d<0 || d>maxdays[m] || y<0 || y>9999) // Performing Date Validation

            {

            System.out.println("The day, month or year are outside acceptable limit");

            }

            else

            {

                /\* First Part \*/

                System.out.println("Date in dd/mm/yyyy format = "+dd+"/"+mm+"/"+yy);

                /\* Second Part \*/

                System.out.print("Date in dd, month name, yyyy format = "+dd+" "+month[m]+", "+yy);

            }

        }

        else

            System.out.println("Wrong Input");

    }

}

#### **Output:**

#### **1)**Enter any date in 8 digits (ddmmyyyy) format: 02052013 Date in dd/mm/yyyy format = 02/05/2013 Date in dd, month name, yyyy format = 02 May, 2013

**2)**Enter any date in 8 digits (ddmmyyyy) format: 12111963  
Date in dd/mm/yyyy format = 12/11/1963  
Date in dd, month name, yyyy format = 12 November, 1963

**3)**Enter any date in 8 digits (ddmmyyyy) format: 252013  
Wrong Input

**4)**Enter any date in 8 digits (ddmmyyyy) format: 29022013  
The day, month or year are outside acceptable limit

#### Question36:

Write a program to input a natural number less than 1000 and display it in words. [Note we have solved the program for numbers in the range [1-9999]

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

**Sample 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.io.\*;

class Num2Word\_ISC2011

{

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

       {

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

        String ty[]={"","","Twenty","Thirty","Forty","Fifty","Sixty","Seventy","Eighty","Ninety"};

        String ten[]={"","Ten","Eleven","Twelve","Thirteen","Fourteen","Fifteen","Sixteen","Seventeen",

                      "Eighteen","Nineteen"};

        String unit[]={"","One","Two","Three","Four","Five","Six","Seven","Eight","Nine"};

        System.out.print("Enter a Number : ");

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

        /\*checking whether the number is in the range [1-9999] or not\*/

        if(n<1 || n>9999)

            System.out.println("Out of Range");

        else

        {

        int th=n/1000; //finding the digit at thousand's place

        int h=(n/100)%10; //finding the digit at hundred's place

        int t=(n/10)%10; //finding the digit at ten's place

        int u=n%10; //finding the digit at unit's place

        System.out.print("Output = ");

        /\*Condition for printing digit at thousand's place, is that it should not be zero\*/

        if(th!=0)

        System.out.print(unit[th]+" Thousand");

        /\*Condition for printing digit at hundred's place, is that it should not be zero\*/

        if(h!=0)

        System.out.print(" "+unit[h]+" Hundred");

        /\*Condition for printing the word "And"\*/

        if((t!=0 || u!=0)&&(th!=0 || h!=0))

        System.out.print(" And");

        /\*Condition for printing digit at ten's place\*/

        if(t==1) //When digit at ten's place is 1, we have different words like Ten, Eleven etc.

        System.out.print(" "+ten[u+1]);

        else //if it is not 1 then we print the words following a normal pattern

        System.out.print(" "+ty[t]+" "+unit[u]);

        }

    }

}

#### Output:

**1.** Enter a Number : 129  
Output = One Hundred And Twenty Nine

**2.** Enter a Number : 8307  
Output = Eight Thousand Three Hundred And Seven

**3.** Enter a Number : 54987  
Out of Range

#### ****Question37:****

Write a Program in Java to input a number and check whether it is a **Keith Number** or not.

**Note:**A Keith Number is an integer N with ‘d’ digits with the following property:  
  
If a Fibonacci-like sequence (in which each term in the sequence is the sum of the ‘d’ previous terms) is formed, with the first ‘d’ terms being the decimal digits of the number N, then N itself occurs as a term in the sequence.  
  
For example, 197 is a Keith number since it generates the sequence  
1, 9, 7, 17, 33, 57, 107, 197, ………..  
  
Some keith numbers are: 14 ,19, 28 , 47 , 61, 75, 197, 742, 1104, 1537……………

import java.io.\*;

class Keith

{

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

    {

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

     System.out.print("Enter the number : "); //inputting the number

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

     int copy=n;

     String s=Integer.toString(n);

     int d=s.length(); //finding the number of digits (d) in the number

     int arr[]=new int[n]; //array for storing the terms of the series

     for(int i=d-1; i>=0; i--)

     {

         arr[i]=copy%10; //storing the digits of the number in the array

         copy=copy/10;

     }

     int i=d,sum=0;

     while(sum<n) //finding the sum till it is less than the number

     {

         sum = 0;

         for(int j=1; j<=d; j++) //loop for generating and adding the previous 'd' terms

         {

             sum=sum+arr[i-j];

         }

         arr[i]=sum; //storing the sum in the array

         i++;

     }

     /\* When the control comes out of the while loop, either the

        sum is equal to the number or greater than it \*/

     if(sum==n) //if sum is equal to the number, then it is a Keith number

        System.out.println("The number is a Keith Number");

     else

        System.out.println("The number is a not a Keith Number");

    }

}

#### **Output:**

Enter the number : 197  
The number is a Keith Number

Enter the number : 14  
The number is a Keith Number

Enter the number : 53  
The number is a not a Keith Number

#### Question38:

A positive natural number, (for e.g. 27), can be represented as follows:  
2+3+4+5+6+7  
8+9+10  
13+14  
where every row represents a combination of consecutive natural numbers, which add up to 27.

Write a program which inputs a positive natural number N and prints the possible consecutive number combinations, which when added give N.

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

**SAMPLE DATA**

**INPUT:**N = 9

**OUTPUT:**4 + 5  
2 + 3+ 4

import java.io.\*;

class ISC06PQ1

    {

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

        {

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

            System.out.print("Enter a number : "); //inputting the number

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

            int sum=0,c=0,j=0;

            for(int i=1;i<n;i++)

            {

                sum=i;

                j=i+1;

/\* adding consecutive natural numbers till sum is less than the given number \*/

                while(sum<n)

                {

                    sum=sum+j;

                    j++;

                }

/\* when the above while condition is false, then either the sum is equal to

that number or is greater than that number

So, we will be printing the series of consecutive numbers only if the sum is

equal to that number

Note: variable 'i' is keeping record of the first number of the series and

variable 'j' is keeping record of the last number of the series \*/

                    if(sum==n)

                    {

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

                        {

                            if(k==i)

                                System.out.print(k);

                            else

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

                        }

                     System.out.println();

                    }

            }

        }

    }

#### Output:

**1.** Enter a number : 15

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

**2.** Enter a number : 95

5 + 6 + 7 + 8 + 9 + 10 + 11 + 12 + 13 + 14  
17 + 18 + 19 + 20 + 21  
47 + 48

#### Question39:

Write a program to input a word from the user and remove the consecutive repeated characters by replacing the sequence of repeated characters by its single occurrence.

**Example:**

INPUT – Jaaavvvvvvvvaaaaaaaaaaa  
OUTPUT – Java

INPUT – Heeeiiiissggoiinggg  
OUTPUT – Heisgoing

import java.io.\*;

class RemoveRepChar

{

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

    {

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

        System.out.print("Enter any word: "); // Inputting the word

        String s = br.readLine();

        s = s + " "; // Adding a space at the end of the word

        int l=s.length(); // Finding the length of the word

        String ans=""; // Variable to store the final result

        char ch1,ch2;

        for(int i=0; i<l-1; i++)

        {

            ch1=s.charAt(i); // Extracting the first character

            ch2=s.charAt(i+1); // Extracting the next character

// Adding the first extracted character to the result if the current and the next characters are different

            if(ch1!=ch2)

            {

            ans = ans + ch1;

            }

        }

        System.out.println("Word after removing repeated characters = "+ans); // Printing the result

    }

}

#### Output:

**Example 1:**  
Enter any word: Jaaavvvvvvvvaaaaaaaaaaa  
Word after removing repeated characters = Java

**Example 2:**  
Enter any word: iiiiiisssssfffffffffffffuunnnnn  
Word after removing repeated characters = isfun

#### Question 40;

#### Convert the 2D array into 1D and then sort that 1D array.

import java.io.\*;

class Sort2D\_Method1

    {

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

        {

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

            System.out.print("Enter the no. of  rows: "); //inputting number of rows

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

            System.out.print("Enter the no. of columns: "); //inputting number of columns

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

            int A[][]=new int[m][n]; //creating a 2D array

            /\* Inputting the 2D Array \*/

            for(int i=0;i<m;i++)

            {

                for(int j=0;j<n;j++)

                {

                    System.out.print("Enter the elements: ");

                    A[i][j]=Integer.parseInt(br.readLine());

                }

            }

            /\* Printing the original 2D Array \*/

            System.out.println("The original array:");

            for(int i=0;i<m;i++)

            {

                for(int j=0;j<n;j++)

                {

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

                }

                System.out.println();

            }

            /\* Saving the 2D Array into a 1D Array \*/

            int B[]=new int[m\*n]; //creating a 1D Array of size 'r\*c'

            int x = 0;

            for(int i=0;i<m;i++)

            {

                for(int j=0;j<n;j++)

                {

                    B[x] = A[i][j];

                    x++;

                }

            }

            /\*Sorting the 1D Array in Ascending Order\*/

            int t=0;

            for(int i=0; i<(m\*n)-1; i++)

            {

                for(int j=i+1; j<(m\*n); j++)

                {

                    if(B[i]>B[j])

                    {

                        t=B[i];

                        B[i]=B[j];

                        B[j]=t;

                    }

                }

            }

            /\*Saving the sorted 1D Array back into the 2D Array \*/

            x = 0;

            for(int i=0;i<m;i++)

            {

                for(int j=0;j<n;j++)

                {

                    A[i][j] = B[x];

                    x++;

                }

            }

            /\* Printing the sorted 2D Array \*/

            System.out.println("The Sorted Array:");

            for(int i=0;i<m;i++)

            {

                for(int j=0;j<n;j++)

                {

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

                }

                System.out.println();

            }

        }

#### }

#### Question41:

Write a program to input a list of integers in an array and arrange them in a way similar to the to-and-fro movement of a Pendulum.

The minimum element out of the list of integers, must come in center position of array. The number in the ascending order next to the minimum, goes to the left, the next higher number goes to the right of minimum number and it continues. As higher numbers are reached, one goes to either side of the minimum value in a to-and-fro manner similar to that of a Pendulum.

**Example:**

INPUT – 1   2   3   4   5  
OUTPUT – 5   3   1   2   4

INPUT – 11   12   31   14   5

OUTPUT – 31   12   5   11   14

import java.io.\*;

class Pendulum\_Array

{

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

    {

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

        System.out.print("nEnter number of elements: "); // Inputting the number of elements

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

        int A[]=new int[n]; //original array

        int B[]=new int[n]; //array for storing the result

        /\*Inputting the Array\*/

        for(int i=0; i<n; i++)

        {

            System.out.print("Enter Element "+(i+1)+": ");

            A[i] = Integer.parseInt(br.readLine());

        }

        /\*Sorting the Inputted Array in Ascending Order\*/

        int t=0;

        for(int i=0; i<n-1; i++)

        {

            for(int j=i+1; j<n; j++)

            {

                if(A[i]>A[j])

                {

                    t=A[i];

                    A[i]=A[j];

                    A[j]=t;

                }

            }

        }

        /\*Printing the Sorted Array\*/

        System.out.println("\nThe Sorted Array Is");

        for(int i=0; i<n; i++)

        {

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

        }

         int mid = (n-1)/2; //finding index of middle cell

         int x = 1, lim = n-1-mid;

         /\*'x' is for accessing elements of array A[] and

         'lim' is for the number of times we have to make this to-and-fro movement\*/

         /\* Pendulum Arrangement Starts Here \*/

         B[mid]=A[0]; //putting the minimum element in the middle cell

         for(int i=1; i<=lim; i++)

         {

             if((mid+i)<n) //going to the right side

                B[mid+i]=A[x++];

             if((mid-i)>=0) //going to the left side

                B[mid-i]=A[x++];

         }

        /\*Printing the Result\*/

        System.out.println("\n\nThe Result Is");

        for(int i=0; i<n; i++)

        {

            System.out.print(B[i]+"\t");

        }

    }

}

#### Output:

**Example 1:**  
Enter number of elements: 5  
Enter Element 1: 1  
Enter Element 2: 2  
Enter Element 3: 3  
Enter Element 4: 4  
Enter Element 5: 5

The Sorted Array Is  
1   2   3   4   5

The Result Is  
5   3   1   2   4

**Example 2:**  
Enter number of elements: 6  
Enter Element 1: 12  
Enter Element 2: 34  
Enter Element 3: 11  
Enter Element 4: 5  
Enter Element 5: 67  
Enter Element 6: 20

The Sorted Array Is  
5   11   12   20   34   67

The Result Is  
34   12   5   11   20   67

#### Question42:

A simple encryption system uses a shifting process to hide a message. The value of the shift can be in the range 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

Fist an extra space is added to the end of the string. To make things 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:**

**1.**INPUT:  
CODED TEXT : “UHINBY LKKQCH HYLKK”  
SHIFT : 7  
OUTPUT:  
DECODED TEXT : ANOTHER VALUE

**2.**INPUT:  
CODED TEXT : “RUIJGG EVGGBK SAGG”  
SHIFT : 11  
OUTPUT:  
DECODED TEST : BEST OF LUCK

**3.**INPUT:  
CODED TEXT : “DKSMMW NAMMUK QMM”  
SHIFT : 29  
OUTPUT:  
INVALID SHIFT VAULE

import java.io.\*;

public class Decode\_ISC2003

{

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

    {

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

        System.out.print("Enter Coded Text : "); // inputting coded text

        String s = br.readLine();

        int l = s.length();

        s = s.toUpperCase(); // converting the coded text into Uppercase

        s = s + " "; // adding a space at the end

        if(l>=100) // checking whether length of inputted code is less than 100

            System.out.println("!!! Invalid Length of Coded Text !!!");

        else

        {

            System.out.print("Enter the Shift Value : ");

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

            if(shift<1 || shift>26) // checking whether shift value is between 1 and 26

                System.out.println("!!! Invalid Shift Value !!!");

            else

            {

                int a, b;

                char ch1, ch2;

                String dec=""; //new String for storing the decoded text

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

                {

                    ch1 = s.charAt(i); // extracting characters one by one

                    ch2 = s.charAt(i+1); // extracting the next character

                    /\* Below we are adding shift value to the characters

                     \* if ch1 = 'A' and shift = 7,

                     \* then ch1 + shift - 1 will give us: 'A'+7-1 = 65+7-1 = 71

                     \* which is the ASCII value of 'G'

                     \*/

                    a = ch1 + shift - 1; // storing ASCII values after adding shift to the current character

                    b = ch2 + shift - 1; // storing ASCII values after adding shift to the next character

                    /\* If the currrent character and the next character are both 'Q' then we have a 'space'

                     \* hence the ASCII value should be 32

                     \*/

                    if((char)a == 'Q' && (char)b == 'Q')

                    {

                        a = 32;

                        i++;

                    }

                    /\* If ASCII value after adding the shift becomes more than 90,

                     \* then we subtract 26 from it, to make it circular,

                     \* eg. 'U'+7-1 = 85+7-1 = 91, but we want 'A' whose ASCII value is 65

                     \* so 91-26 will give us 65

                     \*/

                    if(a>90)

                        a = a - 26;

                    if(ch1 != ' ')

                        dec = dec + (char)a; // finally adding the decoded character to the new String

                }

            System.out.println("Decoded Text : "+dec);

            }

        }

    }

}

#### Output:

**Example 1:**  
Enter Coded Text : UHINBYLKKQCHHYLKK  
Enter the Shift Value : 7  
Decoded Text : ANOTHER WINNER

**Example 2:**  
Enter Coded Text : RUIJGGEVGGBKSAGG  
Enter the Shift Value : 11  
Decoded Text : BEST OF LUCK

**Example 3:**  
Enter Coded Text : UHINBYLKKQCHHYLKK  
Enter the Shift Value : 27  
Invalid Shift Value.

#### Question43:

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 above.

**SAMPLE DATA :**

**1.** INPUT :  
TIME : 3,0  
OUTPUT : 3 : 00 Three o’ clock

import java.io.\*;

public class TimeInWords\_ISC2003

{

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

{

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

/\* Inputting hours and minutes \*/

System.out.print("Enter Hours : ");

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

System.out.print("Enter Minutes : ");

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

if((h>=1 && h<=12) && (m>=0 && m<=59)) // checking whether given input is legal or not.

{

/\* creating an array containing numbers from 1-29 in words \*/

String words[]={"", "One", "Two", "Three", "Four", "Five", "Six","Seven", "Eight", "Nine","Ten",

"Eleven","Twelve","Thirteen","Fourteen","Fifteen","Sixteen","Seventeen","Eighteen","Nineteen",

"Twenty","Twenty one", "Twenty two", "Twenty three", "Twenty four", "Twenty five",

"Twenty six","Twenty seven","Twenty eight", "Twenty nine"};

/\* The below code is for finding whether to print the word 'minute' or 'minutes' \*/

String plu, a;

if(m == 1 || m == 59)

plu = "Minute";

else

plu = "Minutes";

/\* When we have minutes from 31-59, we print the hour ahead of the given hour

\* like 6:55 will be 5 minutes to 7 and not 5 minutes to 6

\* when we print the hour ahead of the given hour, we face a problem at hour = 12

\* because if we print an hour ahead of 12, it will be thirteen, but we want 1

\* so the below code checks this & decides what hour to print in words when minutes is from 31-59

\*/

if(h==12)

a = words[1]; //storing 'one' when hour is 12

else

a = words[h+1]; //if hour is not 12, then storing in words, an hour ahead of given hour

/\* The below code checks minutes and accordingly prints the time in words using array. \*/

System.out.print("Output : "+h+":"+m+" ----- "); //printing the given time in numbers

if(m==0)

System.out.println(words[h]+" O' clock");

else if(m==15)

System.out.println("Quarter past "+words[h]);

else if(m==30)

System.out.println("Half past "+words[h]);

else if(m==45)

System.out.println("Quarter to "+a);

else if(m<30) // condition for minutes between 1-29

System.out.println(words[m]+" "+plu+" past "+words[h]);

else // condition for minutes between 31-59

System.out.println(words[60-m]+" "+plu+" to "+a);

} //end of outer if

else

System.out.println("Invalid Input !"); //printing error message for illegal input

}

}

#### Output:

**Example 1:**  
Enter Hours : 12  
Enter Minutes : 39  
Output : 12:39 —– Twenty one Minutes to One

**Example 2:**  
Enter Hours : 12  
Enter Minutes : 39  
Output : 12:39 —– Twenty one Minutes to One

#### ****Question44:****

A sentence in the Special Fashion can be printed by taking two integers (not beyond total number of words in the sentence or less than 1). These integers tell the word number of the sentence. Replace only those words present at those given integer places by the next character in a circular fashion according to the English Alphabets. If both the integers are same then replace only one word. let us consider the following examples:

**Input Sentence:** He has good Books.  
**Input Integers:** 2, 4  
**Output Sentence:**He ibt good Cpplt.  
(i.e. word number 2 and 4 have been replaced by the next characters in a circular fashion)

**Input Sentence:** Time and tide waits for none.  
**Input Integers:** 3, 3  
**Output Sentence:**Time and ujef waits for none.

Write a case sensitive program that reads a sentence from console (the characters of the sentence may be capital or small or mixed) and two positive integers and output the same sentence after replacing those words present at those given integer places by the next character in a circular fashion according to the English Alphabets.

In the first example given above, word number 2, i.e. “has” is replaced by next characters and hence it becomes “ibt”. Similarly, word number 4, i.e. “Books” is replaced by next characters and hence it becomes “Cpplt”.

import java.io.\*;

class Special\_Fashion

{

String repChar(String s) //function doing the work of replacing characters

{

int l = s.length(), a = 0;

char ch;

String res="";

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

{

ch = s.charAt(i); //extracting characters one by one

a = ch + 1; //storing ASCII values after adding 1 to the current character

if(ch=='z' || ch=='Z')

{

a = a - 26;

}

res = res + (char)a; //finally adding the changed character to the new String

}

return res;

}

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

{

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

Special\_Fashion ob = new Special\_Fashion();

System.out.print("nEnter any sentence : "); //Inputting the sentence

String s = br.readLine();

String ans=""; //String variable to store the final result

String word[]=s.split("[. ]+"); //saving the words of the sentence in an array using split()

int c = word.length;

System.out.print("Enter the 1st word number : ");

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

System.out.print("Enter the 2nd word number : ");

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

if(x<1 || y<1 || x>c || y>c) //checking whether integers inputted are acceptable or not

{

System.out.println("Sorry! The word numbers inputted are out of range");

}

else

{

if(x != y)

{

word[y-1]=ob.repChar(word[y-1]); //sending the words to the repChar() function

}

word[x-1]=ob.repChar(word[x-1]);

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

{

ans = ans + word[i] + " ";

}

System.out.print("Output = "+ans.trim()+".");

}

}

}

#### **Output:**

**1)**Enter any sentence : I love Java for School.  
Enter the 1st word number : 2  
Enter the 2nd word number : 5  
Output = I mpwf Java for Tdippm.

**2)**Enter any sentence : I love Java for School  
Enter the 1st word number : 4  
Enter the 2nd word number : 4  
Output = I love Java gps School.

**3)**Enter any sentence : I love Java for School  
Enter the 1st word number : 2  
Enter the 2nd word number : 6  
Sorry! The word numbers inputted are out of range

#### Question 45:

Write a program to input a string (word). Convert it into lowercase letters. Count and print the frequency of each alphabet present in the string. The output should be given as:  
**Sample Input:** Alphabets  
**Sample Output:**==========================  
Alphabet             Frequency  
==========================  
a                              2  
b                              1  
e                              1  
h                              1  
l                               1  
p                              1  
s                              1  
t                               1

import java.io.\*;

class AlphabetFreq

{

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

{

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

System.out.print("Enter any string: ");

String s = br.readLine();

s=s.toLowerCase(); //converting the string into lowercase

int l=s.length(); //finding the length of the string

char ch;

System.out.println("Output:");

System.out.println("=========================="); //this is just for styling the look of the output

System.out.println("Alphabet\tFrequency");

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

/\* Counting frequency of alphabets begins below \*/

int count=0;

for(char i='a'; i<='z'; i++)

{

count = 0;

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

{

ch=s.charAt(j); //extracting characters of the string one by one

if(ch==i) //first checking the whole string for 'a', then 'b' and so on

count++; //increasing count of those aplhabets which are present in the string

}

if(count!=0)//printing only those alphabets whose count is not '0'

{

System.out.println(i+"\t\t"+count);

}

}

}

}

#### Output:

Enter any string: ilovejavaforschool

Output:  
==========================  
Alphabet             Frequency  
==========================  
a                               2  
c                               1  
e                              1  
f                               1  
h                              1  
i                                1  
j                                1  
l                                2  
o                               4  
r                                1  
s                               1  
v                               2

#### Question 46:

A super class **Record** has been defined to store the names and ranks of 50 students. Define a sub class **Rank** to find the highest rank along with the name. The details of both classes are given below:

**Class name : Record**

**Data Members / instance variables:**  
**name[ ] :** to store the names of students  
**rnk[ ] :** to store the ranks of students

**Member functions:**  
**Record() :** constructor to initialize data members  
**void readvalues() :** to store names and ranks  
**void display() :** displays the names and the corresponding ranks

**Class name : Rank**

**Data Members / instance variables:**  
**index :** integer to store the index of the topmost rank

**Member functions**  
**Rank() :** constructor to invoke the base class constructor and to initialize index to 0.  
**void highest() :** finds the index location of the topmost rank and stores it in index without sorting the array 6  
**void display() :** displays the name and ranks along with the name having the topmost rank.

Specify the class **Record** giving details of the **constructor()**, **void readvalues()**, **void display()**. Using the concept of inheritance, specify the class **Rank**giving details of **constructor()**, **void highest()** and **void display()**.

The main function and algorithm need not be written.

import java.io.\*;

class Record //superclass

{

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

String name[];

int rnk[];

Record()

{

name = new String[50];

rnk = new int[50];

}

void readvalues()throws IOException

{

System.out.println("\*\*\* Inputting The Names And Ranks \*\*\*");

for(int i=0;i<50;i++)

{

System.out.print("Enter name of student "+(i+1)+" : ");

name[i]=br.readLine();

System.out.print("Enter his rank : ");

rnk[i]=Integer.parseInt(br.readLine());

}

}

void display()

{

System.out.println("NamettRank");

System.out.println("-------tt-------"); //this is just for styling the output. You can skip it !

for(int i=0;i<50;i++)

{

System.out.println(name[i]+"tt"+rnk[i]);

}

}

} //end of superclass Record

class Rank extends Record //subclass

{

int index;

Rank()

{

super(); //invoking the constructor of superclass

index = 0;

}

void highest()

{

int min = rnk[0];

for(int i=0;i<50;i++)

{

if(rnk[i]<min)

{

min = rnk[i];

index = i;

}

}

}

void display()

{

super.display(); //calling the superclass function display()

highest(); //calling the function highest() for finding index of topmost rank

System.out.println("nTop most rank = "+rnk[index]);

System.out.println("Student with topmost rank = "+name[index]);

}

} //end of subclass Rank

public class Question11\_ISC2011 //Class which will contain the main() method and execute the program

{

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

{

Rank ob=new Rank(); //creating object of subclass

ob.readvalues(); //calling radvalues() function of superclass to input the names and ranks

System.out.println("\*\*\* Output \*\*\*");

ob.display(); //calling display() function of subclass

}

}

#### Output:

\*\*\* Inputting The Names And Ranks \*\*\*  
Enter name of student 1 : Aamir  
Enter his rank : 5  
Enter name of student 2 : Zakir  
Enter his rank : 2  
Enter name of student 3 : Saalim  
Enter his rank : 7  
Enter name of student 4 : Samir  
Enter his rank : 3  
Enter name of student 5 : Saif  
Enter his rank : 6  
\*\*\* Output \*\*\*  
Name             Rank  
—–                 —–  
Aamir             5  
Zakir              2  
Saalim           7  
Samir             3  
Saif                6

Top most rank = 2  
Student with topmost rank = Zakir

**Question47:**

A super class **Detail** has been defined to store the details of a customer. Define a subclass **Bill**to compute the monthly telephone charge of the customer as per the chart given below:

**Number Of Calls**  
1 – 100  
101 – 200  
201 – 300  
Above 300

**Rate**  
Only Rental charge  
60 paisa per call + rental charge  
80 paisa per call + rental charge  
1 rupee per call + rental charge

The details of both the classes are given below:

**Class Name : Detail**

**Data members / Instance variables:**  
**name :** to store the name of the customer.  
**address :** to store the address of the customer.  
**telno :** to store the phone number of the customer.  
**rent :** to store the monthly rental charge

**Member functions:**  
**Detail(…) :** parameterized constructor to assign values to data members.  
**void show() :** to display the detail of the customer.

**Class Name : Bill**

**Data members / Instance variables:**  
**n :** to store the number of calls.  
**amt :** to store the amount to be paid by the customer.

**Member functions:**  
**Bill(…) :** parameterized constructor to assign values to data members of both classes and to initialize amt = 0.0.  
**void cal() :** calculates the monthly telephone charge as per the charge given above.  
**void show() :** to display the detail of the customer and amount to be paid.

Specify the class **Detail** giving details of the **constructor( )** and **void show()**. Using the concept of inheritance, specify the class **Bill** giving details of the **constructor( )**, **void cal()** and **void show()**.

import java.io.\*;

class Detail //superclass

{

String name, address;

long telno;

double rent;

Detail(String n1, String a1, long t1, double r1)

{

name = n1;

address = a1;

telno = t1;

rent = r1;

}

void show()

{

System.out.println("Name of customer = "+name);

System.out.println("Address = "+address);

System.out.println("Telephone Number = "+telno);

System.out.println("Monthly Rental = Rs. "+rent);

}

} //end of superclass Detail

class Bill extends Detail //subclass

{

int n;

double amt;

Bill(String n1, String a1, long t1, double r1, int c)

{

super(n1,a1,t1,r1); //initializing data members of superclass by calling its constructor

n = c;

amt = 0.0;

}

void cal()

{

if(n>=1 && n<=100)

amt = rent;

else if(n>=101 && n<=200)

amt = 0.6\*n + rent;

else if(n>=201 && n<=300)

amt = 0.8\*n + rent;

else

amt = 1\*n + rent;

}

void show()

{

super.show(); //calling the superclass function show()

System.out.println("No. of calls = "+n);

System.out.println("Amount to be paid = Rs. "+amt);

}

} //end of subclass Bill

/\* In your exams you don't need to write the below given code

We are writing it so as to familiarize the students on how to run

programs based on the concept of inheritance.\*/

public class Question12\_ISC2012 //Class which will contain the main() method and execute the program

{

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

{

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

System.out.print("Enter the name : ");

String n1=br.readLine();

System.out.print("Enter the address : ");

String a1=br.readLine();

System.out.print("Enter the telephone number : ");

long t1=Long.parseLong(br.readLine());

System.out.print("Enter the monthly rental : ");

double r1=Double.parseDouble(br.readLine());

System.out.print("Enter the number of calls : ");

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

Bill ob=new Bill(n1,a1,t1,r1,c); //creating object of subclass

System.out.println("\*\*\* Output \*\*\*");

ob.cal();

ob.show(); //calling show() function of subclass

}

}

#### Output:

Enter the name : Java For School  
Enter the address : 123, Sample Street  
Enter the telephone number : 1234567890  
Enter the monthly rental : 180  
Enter the number of calls : 270  
\*\*\* Output \*\*\*  
Name of customer = Java For School  
Address = 123, Sample Street  
Telephone Number = 1234567890  
Monthly Rental = Rs. 180.0  
No. of calls = 270  
Amount to be paid = Rs. 396.0

#### Question48:

A **Smith number** is a composite number, the sum of whose digits is the sum of the digits of its prime factors obtained as a result of prime factorization (excluding 1). The first few such numbers are 4, 22, 27, 58, 85, 94, 121 ………………..

**Examples:**

1.  666

Prime factors are 2, 3, 3, and 37  
Sum of the digits are (6+6+6) = 18  
Sum of the digits of the factors (2+3+3+(3+7)) = 18

2.   4937775

Prime factors are 3, 5, 5, 65837  
Sum of the digits are (4+9+3+7+7+7+5) = 42  
Sum of the digits of the factors (3+5+5+(6+5+8+3+7)) = 42

Write a program to input a number and display whether the number is a **Smith number** or not.

**Sample data:**

Input             94          Output             SMITH Number

Input             102        Output             NOT SMITH Number

Input             666        Output             SMITH Number

Input             999        Output             NOT SMITH Number

import java.io.\*;

class Smith

{

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

//function for finding sum of digits

int sumDig(int n)

{

int s=0;

while(n>0)

{

s=s+n%10;

n=n/10;

}

return s;

}

//function for generating prime factors and finding their sum

int sumPrimeFact(int n)

{

int i=2, sum=0;

while(n>1)

{

if(n%i==0)

{

sum=sum+sumDig(i); //Here 'i' is the prime factor of 'n' and we are finding its sum

n=n/i;

}

else

i++;

}

return sum;

}

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

{

Smith ob=new Smith();

System.out.print("Enter a Number : ");

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

int a=ob.sumDig(n);// finding sum of digit

int b=ob.sumPrimeFact(n); //finding sum of prime factors

System.out.println("Sum of Digit = "+a);

System.out.println("Sum of Prime Factor = "+b);

if(a==b)

System.out.print("It is a Smith Number");

else

System.out.print("It is Not a Smith Number");

}

}

#### Output:

**1.** Enter a Number : 94  
Sum of Digit = 13  
Sum of Prime Factor = 13  
It is a Smith Number

**2.** Enter a Number : 102  
Sum of Digit = 3  
Sum of Prime Factor = 13  
It is Not a Smith Number

**3.** Enter a Number : 4937775  
Sum of Digit = 42  
Sum of Prime Factor = 42  
It is a Smith Number

#### ****Question49:****

Write a Program in Java to input a number in Decimal number system and convert it into its equivalent number in the Hexadecimal number system.

**Note:** Hexadecimal Number system is a number system which can represent a number in any other number system in terms of digits ranging from 0 to 9 and then A – F only. This number system consists of only sixteen basic digits i.e. 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E and F. Here 10 is represented as A, 11 as B and so on till 15 which is represented as F.

import java.io.\*;

class Dec2Hex

{

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

{

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

System.out.print("Enter a decimal number : ");

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

int r;

String s=""; //variable for storing the result

//array storing the digits (as characters) in a hexadecimal number system

char dig[]={'0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F'};

while(n>0)

{

r=n%16; //finding remainder by dividing the number by 16

s=dig[r]+s; //adding the remainder to the result

n=n/16;

}

System.out.println("Output = "+s);

}

}

#### ****Output:****

Enter a decimal number : 47  
Output = 2F

Enter a decimal number : 1243  
Output = 4DB

#### Question50:

A class **Recursion** has been defined to find the Fibonacci series upto a limit. Some of the members of the class are given below:

**Class Name :** Recursion

**Data Members/instance variables :** a, b, c, limit (all integers)

**Member functions/methods :**

**Recursion() :** constructor to assign a,b,c with appropriate values.  
**void input() :** to accept the limit of the series.  
**int fib(int n) :** to return the **nth** Fibonacci term using**recursive technique**.  
**void genearate\_fibseries() :** to generate the Fibonacci series upto the given limit.

Specify the class **Recursion** giving details of the **constructor**, **int fib()** , **void generate\_fibseries()**. You may assume other functions are written for you and you need not write the main function.

import java.io.\*;

class Recursion

{

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

    int a,b,c,limit;

Recursion() //Constructor

    {

     a=0;

     b=1;

     c=0;

     limit=0;

    }

void input()throws IOException //Function to input the limit

    {

     System.out.print("Enter the limit : ");

     limit=Integer.parseInt(br.readLine());

    }

int fib(int n) //Recursive function generating the 'nth' term of Fibonacci Series

    {

    if(n<=1)

    return a;

    else if(n==2)

    return b;

    else

    return (fib(n-1)+fib(n-2));

    }

void generate\_fibseries() //Function generating all the Fibonacci Series numbers upto 'n' terms

    {

        System.out.println("The Fibonacci Series is:");

        for(int i=1;i<=limit;i++)

        {

            c=fib(i);

            System.out.print(c+"  ");

        }

    }

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

  {

   Recursion ob=new Recursion();

   ob.input();

   ob.generate\_fibseries();

  }

}

**Output:**

Enter the limit : 11  
The Fibonacci Series is:  
0 1 1 2 3 5 8 13 21 34 55

Enter the limit : 20  
The Fibonacci Series is:  
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_THE END\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_