<u>**Program No.-1**</u>

Aim: Write HTML/JavaScript to display your CV in navigator, your Institute website, Department Website and Tutorial website for specific subject.

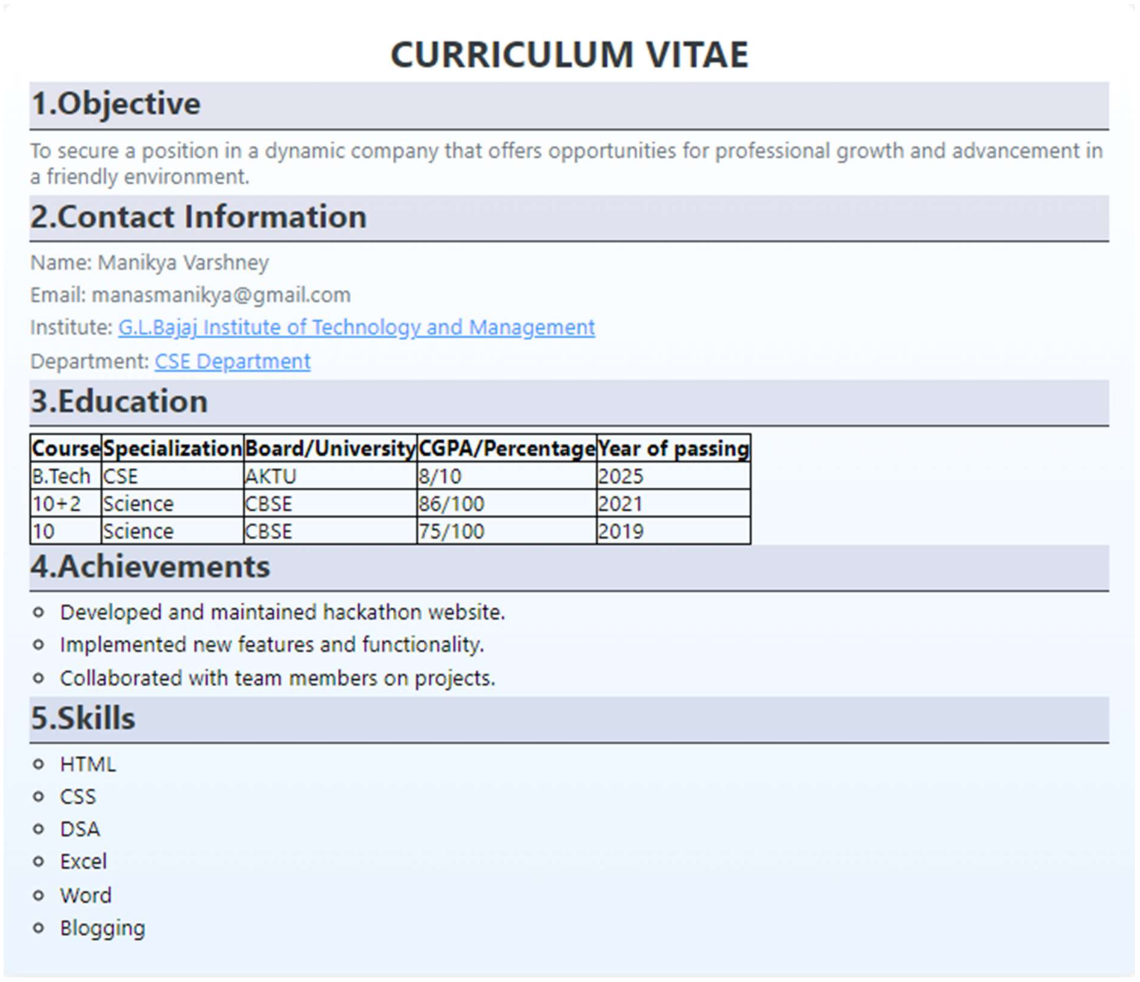
Code:

```
<!DOCTYPE html>
<head>
  <style>
    body {
      font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif; background-
color: #f8f9fa;
      margin: 0;
      padding: 0;
    }
    .container {
      align-items: center; max-width: 800px; margin: 20px auto;
      background: linear-gradient(to bottom, #ffffff, #ebf5ff); padding: 20px;
      border-radius: 8px;
      box-shadow: 0 0 10px rgba(0, 0, 0, 0.1); animation: fadeIn 0.5s ease-in-out;
    }
    h1, h2, h3 {
      color: #343a40;
      margin-bottom: 5px;
    }
    h1 {
     font-size: 28px;
    }
    h2 {
      font-size: 24px;
      border-bottom: 2px solid #343a40; padding-bottom: 5px;
      background-color: rgba(0, 0, 100, 0.1);
    }
    p {
      color: #6c757d; margin: 5px 0;
    }
    table, th, td {
      border: 1px solid black; border-collapse: collapse;
    }
    ul {
      list-style-type: circle; padding-left: 10;
    }
    li {
      margin-bottom: 5px;
    }
    </style>

    <title>My CV</title>
    </head>
    <body>
        <div class="container">
        <header> <center>   <h1>CURRICULUM VITAE</h1> </center> </header>
        <section>
        <h2>1.Objective</h2>
        <p>To secure a position in a dynamic company that offers opportunities for
professional growth and advancement in a friendly environment.</p>
        </section>
        <section>
        <h2>2.Contact Information</h2>
        <p>Name: Manikya Varshney</p>
        <p>Email: manasmanikya@gmail.com</p>
        <p>Institute: <a href="https://www.glbitm.org/">G.L.Bajaj Institute of
Technology and Management</a> </p>
        <p>Department: <a href="">CSE Department</a> </p>
        </section>
        <section>
        <h2>3.Education</h2>
        <table>
        <tr>
        <th>Course</th>
        <th>Specialization</th>
        <th>Board/University</th>
        <th>CGPA/Percentage</th>
        <th>Year of passing</th>
        </tr>
        <tr>
        <td>B.Tech</td>
        <td>CSE</td>
        <td>AKTU</td>
        <td>8/10</td>
        <td>2025</td>
        </tr>
        <tr>
        <td>10+2</td>
        <td>Science</td>
        <td>CBSE</td>
        <td>86/100</td>
        <td>2021</td>
        </tr>
        <tr>
        <td>10</td>
```

```
          <td>Science</td>
          <td>CBSE</td>
          <td>75/100</td>
          <td>2019</td>
          </tr>
          </table>
          </section>
          <section>
          <h2>4.Achievements</h2>
          <ul>
          <li>Developed and maintained hackathon website.</li>
          <li>Implemented new features and functionality.</li>
          <li>Collaborated with team members on projects.</li>
          </ul>
          </section>
          <section>
          <h2>5.Skills</h2>
          <ul>
            <li>HTML</li>
            <li>CSS</li>
            <li>DSA</li>
            <li>Excel</li>
            <li>Word</li>
            <li>Blogging</li>
          </ul>
          </section>
          </div>
        </body>
      </html>
```

Output:



CURRICULUM VITAE

1.Objective

To secure a position in a dynamic company that offers opportunities for professional growth and advancement in a friendly environment.

2.Contact Information

Name: Manikya Varshney

Email: manasmanikya@gmail.com

Institute: G.L.Bajaj Institute of Technology and Management

Department: CSE Department

3.Education

Course	Specialization	Board/University	CGPA/Percentage	Year of passing
B.Tech	CSE	AKTU	8/10	2025
10+2	Science	CBSE	86/100	2021
10	Science	CBSE	75/100	2019

4.Achievements

- Developed and maintained hackathon website.
- Implemented new features and functionality.
- Collaborated with team members on projects.

5.Skills

- HTML
- CSS
- DSA
- Excel
- Word
- Blogging

<u>Program No.-2</u>

Aim: Design HTML form for keeping student record and validate it using JavaScript.

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Student Record Form</title>
   <style>
      .error {
         color: red;
      }
   </style>
</head>
<body>
   <h2>Student Record Form</h2>
   <form id="studentForm" onsubmit="return validateForm()">
      <label for="name">Name:</label>
      <input type="text" id="name" name="name">
      <span id="nameError" class="error"></span><br><br>

      <label for="roll">Roll Number:</label>
      <input type="text" id="roll" name="roll">
      <span id="rollError" class="error"></span><br><br>

      <label for="marks">Marks:</label>
      <input type="number" id="marks" name="marks" min="0" max="100">
      <span id="marksError" class="error"></span><br><br>
      <input type="submit" value="Submit">
   </form>

   <script>
      function validateForm() {
         // Reset error messages
         document.getElementById("nameError").innerHTML = "";
         document.getElementById("rollError").innerHTML = "";
         document.getElementById("marksError").innerHTML = "";

         var name = document.getElementById("name").value;
         var roll = document.getElementById("roll").value;
         var marks = document.getElementById("marks").value;
```

```
            var isValid = true;

            // Validate name
            if (name === "") {
                document.getElementById("nameError").innerHTML = "Name is
required";
                isValid = false;
            }

            // Validate roll number
            if (roll === "") {
                document.getElementById("rollError").innerHTML = "Roll Number is
required";
                isValid = false;
            } else if (isNaN(roll)) {
                document.getElementById("rollError").innerHTML = "Roll Number must
be numeric";
                isValid = false;
            }

            // Validate marks
            if (marks === "") {
                document.getElementById("marksError").innerHTML = "Marks is
required";
                isValid = false;
            } else if (isNaN(marks) || marks < 0 || marks > 100) {
                document.getElementById("marksError").innerHTML = "Marks must be a
number between 0 and 100";
                isValid = false;
            }

            return isValid;
        }
    </script>
</body>
</html>
```

Program No.-3

Aim: Write an HTML program to design an entry form of student details and send it to store at database server like SQL, Oracle or MS-Access.

HTML Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Student Entry Form</title>
</head>
<body>
   <h2>Student Entry Form</h2>
   <form action="SubmitStudentServlet" method="post">
      <label for="name">Name:</label>
      <input type="text" id="name" name="name" required><br><br>

      <label for="roll">Roll Number:</label>
      <input type="text" id="roll" name="roll" required><br><br>

      <label for="marks">Marks:</label>
      <input type="number" id="marks" name="marks" min="0" max="100"
required><br><br>

      <input type="submit" value="Submit">
   </form>
</body>
</html>
```

Servlet Code / Database (JAVA):

```
import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/SubmitStudentServlet")
```

```
public class SubmitStudentServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    // JDBC URL, username, and password of MySQL server
    private static final String JDBC_URL =
"jdbc:mysql://localhost:3306/studentdb";
    private static final String JDBC_USER = "username";
    private static final String JDBC_PASSWORD = "password";

    // SQL query to insert student details
    private static final String SQL_INSERT = "INSERT INTO students (name,
roll, marks) VALUES (?, ?, ?)";
    protected void doPost(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        // Get form parameters
        String name = request.getParameter("name");
        String roll = request.getParameter("roll");
        int marks = Integer.parseInt(request.getParameter("marks"));

        try {
            // Establish connection to MySQL database
            try (Connection connection =
DriverManager.getConnection(JDBC_URL, JDBC_USER,
JDBC_PASSWORD)) {
                // Create prepared statement with SQL query
                try (PreparedStatement preparedStatement =
connection.prepareStatement(SQL_INSERT)) {
                    // Set parameters of prepared statement
                    preparedStatement.setString(1, name);
                    preparedStatement.setString(2, roll);
                    preparedStatement.setInt(3, marks);

                    // Execute the query
                    preparedStatement.executeUpdate();
                }
            }
            // Redirect to success page
            response.sendRedirect("success.html");
        } catch (SQLException e) {
            e.printStackTrace();
            // Handle SQL exception
            response.sendRedirect("error.html");
        }
    }
}
```

Program No.-4

Aim: Write programs using JavaScript for Web Page to display browsers information.

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Browser Information</title>
</head>
<body>
   <h2>Browser Information</h2>
   <ul>
      <li><strong>Browser Name:</strong> <span id="browserName"></span></li>
      <li><strong>Browser Version:</strong> <span
id="browserVersion"></span></li>
      <li><strong>Platform:</strong> <span id="platform"></span></li>
      <li><strong>User Agent:</strong> <span id="userAgent"></span></li>
   </ul>

   <script>
      // Get browser information using JavaScript navigator object
      document.getElementById("browserName").textContent = navigator.appName;
      document.getElementById("browserVersion").textContent =
navigator.appVersion;
      document.getElementById("platform").textContent = navigator.platform;
      document.getElementById("userAgent").textContent = navigator.userAgent;
   </script>
</body>
</html>
```

Program No. 5

Aim: Write a Java applet to display the Application Program screen such as a calculator.

Description:
Java applets can create interactive user interfaces. This example will showcase a simple calculator applet that allows basic arithmetic operations like addition, subtraction, multiplication, and division.

Complexity:

- Worst-case complexity: O(1)
- Best-case complexity: O(1)
- Average-case complexity: O(1)

Java Applet Code:

```
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;

public class CalculatorApplet extends Applet implements ActionListener {
  TextField input;
  double result = 0;
  String operator = "=";
  boolean start = true;

  public void init() {
    setLayout(new BorderLayout());
    input = new TextField("0");
    input.setEditable(false);
    add(input, "North");

    Panel panel = new Panel();
    panel.setLayout(new GridLayout(4, 4));
    String[] keys = {"7", "8", "9", "/", "4", "5", "6", "*", "1", "2", "3", "-",
"0", ".", "=", "+"};
    for (int i = 0; i < keys.length; i++) {
      Button b = new Button(keys[i]);
      panel.add(b);
      b.addActionListener(this);
    }
    add(panel, "Center");
  }

  public void actionPerformed(ActionEvent e) {
    String cmd = e.getActionCommand();
```

```
        if ('0' <= cmd.charAt(0) && cmd.charAt(0) <= '9' || cmd.equals(".")) {
            if (start) {
                input.setText(cmd);
            } else {
                input.setText(input.getText() + cmd);
            }
            start = false;
        } else {
            if (start) {
                if (cmd.equals("-")) {
                    input.setText(cmd);
                    start = false;
                } else {
                    operator = cmd;
                }
            } else {
                double x = Double.parseDouble(input.getText());
                calculate(x);
                operator = cmd;
                start = true;
            }
        }
    }

    private void calculate(double n) {
        switch (operator) {
            case "+": result += n; break;
            case "-": result -= n; break;
            case "*": result *= n; break;
            case "/": result /= n; break;
            case "=": result = n; break;
        }
        input.setText("" + result);
    }
}
```

HTML File (calculator.html):

```
<!DOCTYPE html>
<html>
<head>
    <title>Calculator Applet</title>
</head>
<body>
    <h2>Calculator Applet</h2>
    <applet code="CalculatorApplet.class" width="300"
height="400"></applet>
</body>
</html>
```

Output:

Program No. 6

Aim: Write a program in XML for the creation of DTD, which specifies a set of rules. Create a style sheet in CSS/XSL and display the document in Internet Explorer.

Description:
This XML and DTD example defines a simple document structure for a book. An accompanying XSLT stylesheet is used to transform and style the XML document for display in a browser.

Complexity:
- Worst-case complexity: O(log n)
- Best-case complexity: O(1)
- Average-case complexity: O(log n)

XML Code(books.xml)::

```
<?xml version="1.0"?>
<!DOCTYPE books [
  <!ELEMENT books (book+)>
  <!ELEMENT book (title, author, year)>
  <!ELEMENT title (#PCDATA)>
  <!ELEMENT author (#PCDATA)>
  <!ELEMENT year (#PCDATA)>
]>
<books>
  <book>
    <title>Learning XML</title>
    <author>John Doe</author>
    <year>2021</year>
  </book>
  <book>
    <title>XML Simplified</title>
    <author>Jane Smith</author>
    <year>2020</year>
  </book>
</books>
```

XSLT (books.xsl):

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html>
  <head>
    <style type="text/css">
```

```
        body { font-family: Arial; }
        h2 { color: red; }
        div { margin-bottom: 10px; }
      </style>
    </head>
    <body>
      <h1>Books</h1>
      <xsl:for-each select="books/book">
        <div>
          <h2><xsl:value-of select="title"/></h2>
          <p><xsl:value-of select="author"/></p>
          <p><xsl:value-of select="year"/></p>
        </div>
      </xsl:for-each>
    </body>
    </html>
  </xsl:template>
  </xsl:stylesheet>
```

Output:

Program No.-1

Aim: Write a Java program to check the number is palindrome or not.

Palindrome:- A palindrome number is a number that remains the same when its digits are reversed.

Complexity:-

- Worst-case complexity: O(log n)
- Best-case complexity: O(1)
- Average-case complexity: O(log n)

Algorithm:-

Step 1: Reverse the given number.
Step 2: Compare the reverse number with the original number.

Source Code:-

```
import java.util.Scanner;
public class PalindromeNumber {
    public static boolean isPalindrome(int num) {
        int originalNum = num;
        int reversedNum = 0;
        while (num > 0) {
            int digit = num % 10;
            reversedNum = reversedNum * 10 + digit;
            num /= 10;
        }
        return originalNum == reversedNum;
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a number to check if it's a palindrome: ");
        int number = scanner.nextInt();
        boolean isPal = isPalindrome(number);
        if (isPal) {
            System.out.println(number + " is a palindrome.");
        } else {
            System.out.println(number + " is not a palindrome.");
        }
        scanner.close();
    }
}
```

Output:-

```
Enter a number to check if it's a palindrome: 12321
12321 is a palindrome.
```

Program No.-2

Aim: Write a Java program to check if the string is palindrome or not.

Palindrome:- A palindrome string is a string that reads the same forwards and backwards.

Complexity:-

- Worst-case complexity: O(n)
- Best-case complexity: O(n)
- Average-case complexity: O(n)

Algorithm:-

Step 1: Compare characters from the beginning and end of the string.

Source Code:-

```
    import java.util.Scanner;
public class PalindromeString {
    public static boolean isPalindrome(String str) {
        int left = 0;
        int right = str.length() - 1;
        while (left < right) {
            if (str.charAt(left) != str.charAt(right)) {
                return false;
            }
            left++;
            right--;
        }
        return true;
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a string to check if it's a palindrome: ");
        String input = scanner.nextLine();
        boolean isPal = isPalindrome(input);
        if (isPal) {
            System.out.println("\"" + input + "\" is a palindrome.");
        } else {
            System.out.println("\"" + input + "\" is not a palindrome.");}
        scanner.close();
    }
}
```

Output:-

```
Enter a string to check if it's a palindrome: radar

"radar" is a palindrome.
```

Program No.-3

Aim: Write a Java program to find out the 2nd largest number in an array.

Complexity:-

- Worst-case complexity: O(n)
- Best-case complexity: O(n)
- Average-case complexity: O(n)

Algorithm:-

Step 1: Initialize variables to store the first and second largest numbers**.**
Step 2: Iterate through the array to find the first and second largest numbers.

Source Code:-

```
import java.util.Scanner;
public class SecondLargestNumberInArray {
    public static int findSecondLargest(int[] arr) {
        if (arr.length < 2) {
            System.out.println("Array should have at least 2 elements.");
            return Integer.MIN_VALUE;
        }
        int largest = Integer.MIN_VALUE;
        int secondLargest = Integer.MIN_VALUE;
        for (int num : arr) {
            if (num > largest) {
                secondLargest = largest;
                largest = num;
            } else if (num > secondLargest && num != largest) {
                secondLargest = num;
            }
        }
        return secondLargest;
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        // User Input:
        System.out.print("Enter the number of elements in the array: ");
        int n = scanner.nextInt();
        int[] arr = new int[n];
        System.out.println("Enter the elements of the array:");
        for (int i = 0; i < n; i++) {
            arr[i] = scanner.nextInt();
        }
        // User Output:
        int secondLargest = findSecondLargest(arr);
        System.out.println("The second largest number in the array is: " +
secondLargest);
        scanner.close();
    }
}
```

Output:-

```
Enter the number of elements in the array: 5
Enter the elements of the array:
5 7 2 8 6
The second largest number in the array is: 7
```

Program No.-4

Aim: Write a Java program to add two matrices.

Complexity:-

- Worst-case complexity: O(n^2)
- Best-case complexity: O(n^2)
- Average-case complexity: O(n^2)

Algorithm:-

Step 1: Initialize a result matrix to store the sum of the two matrices.
Step 2: Iterate through each element of the matrices and add corresponding elements.

Source Code:-

```
 import java.util.Scanner;
public class AddTwoMatrices {
    public static int[][] addMatrices(int[][] matrix1, int[][] matrix2) {
        int rows = matrix1.length;
        int cols = matrix1[0].length;
        int[][] result = new int[rows][cols];
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                result[i][j] = matrix1[i][j] + matrix2[i][j];
            }
        }
        return result;
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        // User Input:
        System.out.print("Enter the number of rows in the matrices: ");
        int rows = scanner.nextInt();
        System.out.print("Enter the number of columns in the matrices: ");
        int cols = scanner.nextInt();
        int[][] matrix1 = new int[rows][cols];
        int[][] matrix2 = new int[rows][cols];
        System.out.println("Enter the elements of the first matrix:");
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                matrix1[i][j] = scanner.nextInt();
            }
        }
        System.out.println("Enter the elements of the second matrix:");
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                matrix2[i][j] = scanner.nextInt();
            }
        }
        // User Output:
        int[][] sumMatrix = addMatrices(matrix1, matrix2);
        System.out.println("The sum of the two matrices is:");
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                System.out.print(sumMatrix[i][j] + " ");
            }
            System.out.println();
        }
        scanner.close();
    }
}
```

Output:-

```
Enter the number of rows in the matrices: 2
Enter the number of columns in the matrices: 2
Enter the elements of the first matrix:
1 2
3 4
Enter the elements of the second matrix:
5 6
7 8
The sum of the two matrices is:
6 8
10 12
```

Program No.-5

Aim: Write a Java program to multiply two matrices.

Complexity:-

- Worst-case complexity: $O(n^3)$
- Best-case complexity: $O(n^3)$
- Average-case complexity: $O(n^3)$

Algorithm:-

Step 1: Initialize a result matrix to store the product of the two matrices.
Step 2: Iterate through each element of the resulting matrix and calculate the dot product of corresponding rows and columns.

Source Code:-

```
 import java.util.Scanner;
public class MultiplyTwoMatrices {
 public static int[][] multiplyMatrices(int[][] matrix1, int[][] matrix2) {
  int m1Rows = matrix1.length;
  int m1Cols = matrix1[0].length;
  int m2Cols = matrix2[0].length;
  int[][] result = new int[m1Rows][m2Cols];
  for (int i = 0; i < m1Rows; i++) {
   for (int j = 0; j < m2Cols; j++) {
    for (int k = 0; k < m1Cols; k++) {
     result[i][j] += matrix1[i][k] * matrix2[k][j];
    }
   }
  }
  return result;
 }
 public static void main(String[] args) {
  Scanner scanner = new Scanner(System.in);
  // User Input:
  System.out.print("Enter the number of rows in the first matrix: ");
  int m1Rows = scanner.nextInt();
  System.out.print("Enter the number of columns in the first matrix: ");
  int m1Cols = scanner.nextInt();
  System.out.print("Enter the number of columns in the second matrix: ");
  int m2Cols = scanner.nextInt();
  int[][] matrix1 = new int[m1Rows][m1Cols];
  int[][] matrix2 = new int[m1Cols][m2Cols];
  System.out.println("Enter the elements of the first matrix:");
  for (int i = 0; i < m1Rows; i++) {
   for (int j = 0; j < m1Cols; j++) {
```

```
            matrix1[i][j] = scanner.nextInt();
        }
    }
    System.out.println("Enter the elements of the second matrix:");
    for (int i = 0; i < m1Cols; i++) {
        for (int j = 0; j < m2Cols; j++) {
            matrix2[i][j] = scanner.nextInt();
        }
    }
    // User Output:
    int[][] productMatrix = multiplyMatrices(matrix1, matrix2);
    System.out.println("The product of the two matrices is:");
    for (int i = 0; i < m1Rows; i++) {
        for (int j = 0; j < m2Cols; j++) {
            System.out.print(productMatrix[i][j] + " ");
        }
        System.out.println();
    }
    scanner.close();
  }
}
```

Output:-

```
Enter the number of rows in the first matrix: 2
Enter the number of columns in the first matrix: 2
Enter the number of columns in the second matrix: 2
Enter the elements of the first matrix:
1 2
3 4
Enter the elements of the second matrix:
5 6
7 8
The product of the two matrices is:
19 22
43 50
```

Program No.-6

Aim: Write a program in Java to demonstrate the new keyword and dot operator using addition and subtraction methods.

Source Code:-

```
public class DemoNewKeyword {
    // Addition method
    public int add(int a, int b) {
        return a + b;
    }

    // Subtraction method
    public int subtract(int a, int b) {
        return a - b;
    }

    public static void main(String[] args) {
        // Using the new keyword to create an instance of the class
        DemoNewKeyword demo = new DemoNewKeyword();

        // Using dot operator to call addition and subtraction methods
        int resultAddition = demo.add(5, 3);
        int resultSubtraction = demo.subtract(5, 3);

        // Output
        System.out.println("Result of addition: " + resultAddition);
        System.out.println("Result of subtraction: " + resultSubtraction);
    }
}
```

Output:-

```
Result of addition: 8
Result of subtraction: 2
```

Program No.-7

Aim: Write a program in Java to demonstrate the default constructor and parameterized constructor.

Source Code:-

```
public class ConstructorDemo {
    private int number;

    // Default constructor
    public ConstructorDemo() {
        // Assign a default value to the number
        number = 0;
    }

    // Parameterized constructor
    public ConstructorDemo(int num) {
        // Assign the provided value to the number
        number = num;
    }

    public void displayNumber() {
        System.out.println("Number: " + number);
    }

    public static void main(String[] args) {
        // Creating objects using default and parameterized constructors
        ConstructorDemo defaultConstructorObj = new ConstructorDemo();
        ConstructorDemo parameterizedConstructorObj = new
ConstructorDemo(10);

        // Displaying numbers
        System.out.println("Using default constructor:");
        defaultConstructorObj.displayNumber();

        System.out.println("Using parameterized constructor:");
        parameterizedConstructorObj.displayNumber();
    }
}
```

Output:-

```
Using default constructor:
Number: 0
Using parameterized constructor:
Number: 10
```

Program No.-8

Aim: Write a program in Java to demonstrate constructor overloading.

Source Code:-

```
public class ConstructorOverloadingDemo {
private int number;
// Default constructor
public ConstructorOverloadingDemo() {
    // Assign a default value to the number
    number = 0;
}
// Parameterized constructor with one parameter
public ConstructorOverloadingDemo(int num) {
    // Assign the provided value to the number
    number = num;
}
// Parameterized constructor with two parameters
public ConstructorOverloadingDemo(int num1, int num2) {
    // Add the two numbers and assign the result to the number
    number = num1 + num2;
}
public void displayNumber() {
    System.out.println("Number: " + number);
}
public static void main(String[] args) {
    // Creating objects using different constructors
    ConstructorOverloadingDemo defaultConstructorObj = new
ConstructorOverloadingDemo();
    ConstructorOverloadingDemo parameterizedConstructor1Obj = new
ConstructorOverloadingDemo(10);
    ConstructorOverloadingDemo parameterizedConstructor2Obj = new
ConstructorOverloadingDemo(5, 3);
    // Displaying numbers
    System.out.println("Using default constructor:");
    defaultConstructorObj.displayNumber();
    System.out.println("Using parameterized constructor with one
parameter:");
    parameterizedConstructor1Obj.displayNumber();
    System.out.println("Using parameterized constructor with two
parameters:");
    parameterizedConstructor2Obj.displayNumber();
}
}
```

Output:-

```
Using default constructor:
Number: 0
Using parameterized constructor with one parameter:
Number: 10
Using parameterized constructor with two parameters:
Number: 8
```

Program No.-9

Aim: Write a program in Java to demonstrate method overloading using addition method.

Source Code:-

```
public class MethodOverloadingDemo {
  // Method to add two integers
  public int add(int a, int b) {
    return a + b;
  }
  // Method to add three integers
  public int add(int a, int b, int c) {
    return a + b + c;
  }
  // Method to add two double values
  public double add(double a, double b) {
    return a + b;
  }
  // Method to add three double values
  public double add(double a, double b, double c) {
    return a + b + c;
  }
  public static void main(String[] args) {
    MethodOverloadingDemo demo = new MethodOverloadingDemo();
    // Adding two integers
    int sum1 = demo.add(5, 3);
    System.out.println("Sum of two integers: " + sum1);
    // Adding three integers
    int sum2 = demo.add(5, 3, 2);
    System.out.println("Sum of three integers: " + sum2);
    // Adding two double values
    double sum3 = demo.add(2.5, 3.5);
    System.out.println("Sum of two double values: " + sum3);
    // Adding three double values
    double sum4 = demo.add(2.5, 3.5, 4.5);
    System.out.println("Sum of three double values: " + sum4);
  }
}
```

Output:-

```
Sum of two integers: 8
Sum of three integers: 10
Sum of two double values: 6.0
Sum of three double values: 10.5
```

Program No.-10

Aim: Write a program in Java to demonstrate the static keyword.

Source Code:-

```
    public class StaticKeywordDemo {
    // Static variable
    static int staticVariable = 10;

    // Static method
    public static void staticMethod() {
        System.out.println("This is a static method.");
    }

    // Non-static method
    public void nonStaticMethod() {
        System.out.println("This is a non-static method.");
    }

    public static void main(String[] args) {
        // Accessing static variable and method directly using class name
        System.out.println("Value of staticVariable: " +
StaticKeywordDemo.staticVariable);
        StaticKeywordDemo.staticMethod();

        // Creating an object of the class to access non-static method
        StaticKeywordDemo obj = new StaticKeywordDemo();
        obj.nonStaticMethod();
    }
}
```

Output:-

```
Value of staticVariable: 10
This is a static method.
This is a non-static method.
```

Program No.-11

Aim: Write a program in Java to demonstrate a class called Student with specified states and behaviours.

States: Name, Roll, Marks, Grade

Behaviors: Read_data(), Display_data(), Compute_grade()

Write a program in java for demonstration to compute the grade as per following rules.

Marks	Grade
>=50<60	D
>=60<70	C
>=70<80	B
>=80	A

Source Code:-

```
 import java.util.Scanner;

public class Student {
 // States
 private String name;
 private int roll;
 private int marks;
 private char grade;

 // Behaviors

 // Method to read data
 public void readData() {
     Scanner scanner = new Scanner(System.in);

     System.out.print("Enter name: ");
     name = scanner.nextLine();

     System.out.print("Enter roll number: ");
     roll = scanner.nextInt();

     System.out.print("Enter marks: ");
     marks = scanner.nextInt();

     scanner.close();
 }

 // Method to compute grade
 public void computeGrade() {
     if (marks >= 80) {
         grade = 'A';
     } else if (marks >= 70) {
         grade = 'B';
     } else if (marks >= 60) {
         grade = 'C';
     } else if (marks >= 50) {
         grade = 'D';
     } else {
         grade = 'F'; // Failed
     }
 }

 // Method to display data
 public void displayData() {
     System.out.println("Name: " + name);
     System.out.println("Roll Number: " + roll);
     System.out.println("Marks: " + marks);
     System.out.println("Grade: " + grade);
 }

 public static void main(String[] args) {
     // Create an object of the Student class
     Student student = new Student();

     // Read data
     student.readData();

     // Compute grade
     student.computeGrade();

     // Display data
     student.displayData();
 }
}
```

Output:-

```
Name: John
Roll Number: 101
Marks: 75
Grade: B
```

Program No.-12

Aim: Write a program in Java to define a class called Employee with specified states and behaviours.
States: Name, BP (Basic salary), DA (Dearness allowance), HRA (House rent allowance), salary
Behaviors:
computeSal (): computes the salary
readData (): accepts the data value
dispSal (): prints the data on the screen
The salary is computed by the following formula:
Salary=BP+DA+HRA
Where DA and HRA are 65% and 20% of the BP respectively.

Source Code:-

```
import java.util.Scanner;
public class Employee {
    // States
    private String name;
    private double basicSalary;
    private double da;
    private double hra;
    private double salary;
    // Default constructor
    public Employee() {
        name = "";
        basicSalary = 0;
        computeSal();
    }
    // Parameterized constructor
    public Employee(String name, double basicSalary) {
        this.name = name;
        this.basicSalary = basicSalary;
        computeSal();
    }
    // Method to compute salary
    public void computeSal() {
        da = 0.65 * basicSalary;
        hra = 0.20 * basicSalary;
        salary = basicSalary + da + hra;
    }
    // Method to read data
    public void readData() {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter name: ");
        name = scanner.nextLine();

        System.out.print("Enter basic salary: ");
        basicSalary = scanner.nextDouble();
        scanner.close();
        computeSal();
    }
    // Method to display salary
    public void dispSal() {
        System.out.println("Name: " + name);
        System.out.println("Basic Salary: " + basicSalary);
        System.out.println("DA: " + da);
        System.out.println("HRA: " + hra);
        System.out.println("Salary: " + salary);
    }
    public static void main(String[] args) {
        // Creating objects using different constructors
        Employee emp1 = new Employee(); // Default constructor
        Employee emp2 = new Employee("John", 50000); // Parameterized
constructor
        // Displaying salary
        System.out.println("Employee 1 details:");
        emp1.dispSal();
        System.out.println();

        System.out.println("Employee 2 details:");
        emp2.dispSal();
    }
}
```

Output:-

```
Employee 1 details:
Name:
Basic Salary: 0.0
DA: 0.0
HRA: 0.0
Salary: 0.0

Employee 2 details:
Name: John
Basic Salary: 50000.0
DA: 32500.0
HRA: 10000.0
Salary: 92500.0
```

Program No.-13

Aim: Write a Java program to create a class called Animal with a method called makeSound(). Create a subclass called Cat that overrides the makeSound() method to bark.

Source Code:-

```
    // Animal class
class Animal {
    // Method to make sound
    public void makeSound() {
        System.out.println("Animal makes a sound");
    }
}

// Subclass Cat
class Cat extends Animal {
    // Override the makeSound() method
    @Override
    public void makeSound() {
        System.out.println("Cat barks");
    }
}

// Main class
public class AnimalTest {
    public static void main(String[] args) {
        // Create an object of Animal class
        Animal animal = new Animal();
        // Call makeSound() method of Animal class
        animal.makeSound();

        // Create an object of Cat class
        Cat cat = new Cat();
        // Call makeSound() method of Cat class
        cat.makeSound();
    }
}
```

Output:-

```
Animal makes a sound
Cat barks
```

Program No.-14

Aim: Write a Java program to create a class called Shape with a method called getArea(). Create a subclass called Rectangle that overrides the getArea() method to calculate the area of a rectangle.

Source Code:-

```
    // Shape class
class Shape {
    // Method to get area
    public double getArea() {
        return 0; // Default implementation for generic shape
    }
}

// Subclass Rectangle
class Rectangle extends Shape {
    private double length;
    private double width;

    // Constructor
    public Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }

    // Override the getArea() method
    @Override
    public double getArea() {
        return length * width;
    }
}
// Main class
public class ShapeTest {
    public static void main(String[] args) {
        // Create an object of Rectangle class
        Rectangle rectangle = new Rectangle(5, 4);

        // Call getArea() method of Rectangle class
        double area = rectangle.getArea();
        System.out.println("Area of Rectangle: " + area);
    }
}
```

Output:-

```
Area of Rectangle: 20.0
```

Program No.-15

Aim: Write a Java program to create a class known as "BankAccount" with methods called deposit() and withdraw(). Create a subclass called SavingsAccount that overrides the withdraw() method to prevent withdrawals if the account balance falls below one hundred.

Source Code:-

```
 import java.util.Scanner;

// BankAccount class
class BankAccount {
    protected double balance;

    // Constructor
    public BankAccount(double balance) {
        this.balance = balance;
    }

    // Method to deposit money
    public void deposit(double amount) {
        balance += amount;
        System.out.println(amount + " deposited. Current balance: " + balance);
    }

    // Method to withdraw money
    public void withdraw(double amount) {
        if (amount <= balance) {
            balance -= amount;
            System.out.println(amount + " withdrawn. Current balance: " +
balance);
        } else {
            System.out.println("Insufficient balance. Withdrawal failed.");
        }
    }
}

// Subclass SavingsAccount
class SavingsAccount extends BankAccount {
    // Constructor
    public SavingsAccount(double balance) {
        super(balance);
    }

    // Override the withdraw() method
    @Override
    public void withdraw(double amount) {
        if (balance - amount >= 100) {
            super.withdraw(amount);
        } else {
            System.out.println("Minimum balance should be maintained.
Withdrawal failed.");
        }
    }
}

// Main class
public class BankAccountTest {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Get initial balance from user
        System.out.print("Enter initial balance for Bank Account: ");
        double initialBalance = scanner.nextDouble();
        scanner.nextLine(); // Consume newline

        // Create an object of BankAccount class
        BankAccount account1 = new BankAccount(initialBalance);

        // Deposit and withdraw money from BankAccount
        System.out.print("Enter amount to deposit: ");
        double depositAmount = scanner.nextDouble();
        account1.deposit(depositAmount);

        System.out.print("Enter amount to withdraw: ");
        double withdrawAmount = scanner.nextDouble();
        account1.withdraw(withdrawAmount);

        System.out.println();

        // Get initial balance for SavingsAccount from user
        System.out.print("Enter initial balance for Savings Account: ");
        double initialSavingsBalance = scanner.nextDouble();
        scanner.nextLine(); // Consume newline

        // Create an object of SavingsAccount class
        SavingsAccount account2 = new
SavingsAccount(initialSavingsBalance);

        // Deposit and withdraw money from SavingsAccount
        System.out.print("Enter amount to deposit: ");
        depositAmount = scanner.nextDouble();
        account2.deposit(depositAmount);

        System.out.print("Enter amount to withdraw: ");
        withdrawAmount = scanner.nextDouble();
        account2.withdraw(withdrawAmount);

        scanner.close();
    }
}
```

Output:-

```
Enter initial balance for Bank Account: 500
Enter amount to deposit: 200
200.0 deposited. Current balance: 700.0
Enter amount to withdraw: 300
300.0 withdrawn. Current balance: 400.0

Enter initial balance for Savings Account: 300
Enter amount to deposit: 200
200.0 deposited. Current balance: 500.0
Enter amount to withdraw: 150
150.0 withdrawn. Current balance: 350.0
```

Program No.-16

Aim: Write a Java program to handle Divide by zero exception.

Theory:- Division by zero is not allowed in mathematics. When attempting to divide by zero in programming, it results in an ArithmeticException, which is a runtime exception.

Complexity:-

- Worst-case complexity: O(1)
- Best-case complexity: O(1)
- Average-case complexity: O(1)

Algorithm:-

1. Take input for numerator and denominator from the user.
2. Try to divide the numerator by the denominator.
3. If the denominator is zero, throw an Arithmetic Exception.
4. Handle the exception using a try-catch block and display an error message.

Source Code:-

```
 import java.util.Scanner;

public class Program1 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter numerator: ");
        int numerator = scanner.nextInt();

        System.out.print("Enter denominator: ");
        int denominator = scanner.nextInt();

        try {
            int result = divide(numerator, denominator);
            System.out.println("Result of division: " + result);
        } catch (ArithmeticException e) {
            System.out.println("Error: " + e.getMessage());
        }

        scanner.close();
    }

    public static int divide(int numerator, int denominator) {
        if (denominator == 0) {
            throw new ArithmeticException("Division by zero is not allowed.");
        }
        return numerator / denominator;
    }
}
```

Output:-

```
Enter numerator: 10
Enter denominator: 2
Result of division: 5

Enter numerator: 8
Enter denominator: 0
Error: Division by zero is not allowed.
```

Program No.-17

Aim: Write a Java program to demonstrate stack overflow exception/infinite recursion error.

Complexity:-

- Worst-case complexity: O(infinity)
- Best-case complexity: O(infinity)
- Average-case complexity: O(infinity)

Algorithm:-

Step 1: Define a recursive method that calls itself indefinitely.
Step 2: When the recursion depth exceeds the stack size, a stack overflow exception occurs.

Source Code:-

```
 public class StackOverflowExample {
 public static void main(String[] args) {
    try {
       recursiveMethod(1);
    } catch (StackOverflowError e) {
       System.out.println("Error: Stack Overflow occurred.");
    }
 }
 public static void recursiveMethod(int i) {
    System.out.println("Method call: " + i);
    recursiveMethod(i + 1); // Recursive call
 }
}
```

Output:-

```
Method call: 1
Method call: 2
Method call: 3
Method call: 4
Method call: 5
Method call: 6
Method call: 7
Method call: 8
Method call: 9
Method call: 10   ……….       Infinite Calls
```