IP LAB PROGRAMS

1. Create a web page with the following

a. Cascading style sheets
b. Embedded style sheets
c. Inline style sheets
Use our college information for the web pages.

Aim

To create a web page using CSS (Cascading, Embedded, and Inline) style sheets.

Algorithm

1. **START**
2. Set up HTML structure.
3. Add external CSS and link it with `<link>` tag.
4. Add embedded CSS in `<style>` tag within `<head>`.
5. Use inline CSS directly within HTML elements.
6. **STOP**

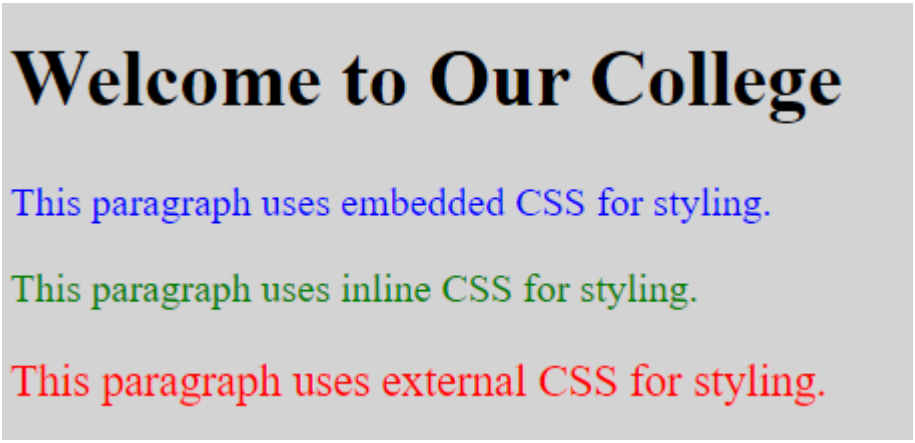
Program (HTML)

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>College Information</title>
    <link rel="stylesheet" href="styles.css" />
    <style>
      body {
        background-color: lightgray;
      }

      .embedded-style {
        color: blue;
      }
    </style>
  </head>

  <body>
    <h1>Welcome to Our College</h1>
    <p class="embedded-style">This paragraph uses embedded CSS for styling.</p>
    <p style="color: green;">This paragraph uses inline CSS for styling.</p>
    <p class="external-style">This paragraph uses external CSS for styling.</p>
  </body>
</html>
```

Program (styles.css)

```css
.external-style {
  color: red;
  font-size: 18px;
  background-color: lightgray;
}
```

Sample Output



Result

Thus, a web page was created successfully using CSS (Cascading, Embedded, and Inline) style sheets.

---

2. Create a web page with the following using HTML to embed a map in a web page, fix the hot spots in that map, and show all the related information when the hot spots are clicked

Aim

To create a web page that embeds an India map with clickable hotspots for important cities, linking to separate HTML files with related information.

Algorithm

1. **START**
2. Set up HTML structure.
3. Embed an India map using the `<img>` tag.
4. Use the `<map>` and `<area>` tags to define hotspots for cities.
5. Link each hotspot to a separate HTML file containing city information.
6. **STOP**

Program (HTML - Main Page)

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>India Map with Cities</title>
    <style>
      body {
        font-family: Arial, sans-serif;
      }
    </style>
  </head>

  <body>
    <h1>India Map</h1>
    <img src="india-map.png" usemap="#india-map" alt="India Map" />
    <map name="india-map">
      <area shape="circle" coords="507,552,24" alt="Delhi" href="delhi.html" />
      <area shape="circle" coords="312,1103,70" alt="Mumbai" href="mumbai.html" />
      <area shape="circle" coords="659,1443,38" alt="Kolkata" href="kolkata.html"
/>
      <area shape="circle" coords="1056,889,36" alt="Chennai" href="chennai.html"
/>
    </map>
  </body>
</html>
```

delhi.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Delhi</title>
  </head>

  <body>
    <h1>Delhi</h1>
    <p>Population: 20 million</p>
    <p>Capital city of India, known for its rich history and culture.</p>
    <a href="index.html">Back to Map</a>
  </body>
</html>
```

mumbai.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Mumbai</title>
  </head>
```

```
  <body>
    <h1>Mumbai</h1>
    <p>Population: 20 million</p>
    <p>Financial capital of India, famous for Bollywood and its beaches.</p>
    <a href="index.html">Back to Map</a>
  </body>
</html>
```

kolkata.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Kolkata</title>
  </head>

  <body>
    <h1>Kolkata</h1>
    <p>Population: 14 million</p>
    <p>Known for its cultural heritage and historic landmarks.</p>
    <a href="index.html">Back to Map</a>
  </body>
</html>
```

chennai.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Chennai</title>
  </head>

  <body>
    <h1>Chennai</h1>
    <p>Population: 7 million</p>
    <p>Known for its temples and cultural landmarks.</p>
    <a href="index.html">Back to Map</a>
  </body>
</html>
```

Sample Output

- An India map with clickable hotspots for Delhi, Mumbai, Kolkata, and Chennai.
- Clicking a hotspot opens a new page with information about the respective city.

Result

Thus, a web page was created successfully that embeds an India map with hotspots linking to separate HTML files for city information.

---

3. Write an XML program and process it using XSLT

Aim

To create an XML document and transform it into an HTML format using XSLT.

Algorithm

1. **START**
2. Create an XML document with data.
3. Create an XSLT stylesheet to transform the XML into HTML.
4. Apply the XSLT to the XML using a processor.
5. **STOP**

Program (XML)

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="library.xsl"?>
<library>
    <book>
        <title>XML Fundamentals</title>
        <author>John Doe</author>
        <year>2022</year>
    </book>
    <book>
        <title>XSLT Essentials</title>
        <author>Jane Smith</author>
        <year>2023</year>
    </book>
    <book>
        <title>Learning XPath</title>
        <author>Emily Johnson</author>
        <year>2021</year>
    </book>
</library>
```

Program (XSLT)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:template match="/library">
        <html>
            <head>
                <title>Library Books</title>
```

```
            </head>
            <body>
                <h1>Book List</h1>
                <table border="1">
                    <tr>
                        <th>Title</th>
                        <th>Author</th>
                        <th>Year</th>
                    </tr>
                    <xsl:for-each select="book">
                        <tr>
                            <td>
                                <xsl:value-of select="title"/>
                            </td>
                            <td>
                                <xsl:value-of select="author"/>
                            </td>
                            <td>
                                <xsl:value-of select="year"/>
                            </td>
                        </tr>
                    </xsl:for-each>
                </table>
            </body>
        </html>
    </xsl:template>
</xsl:stylesheet>
```

Sample Output

Book List

Title	Author	Year
XML Fundamentals	John Doe	2022
XSLT Essentials	Jane Smith	2023
Learning XPath	Emily Johnson	2021

Result

Thus, an XML document was created and processed using XSLT to generate an HTML representation of the data.

---

4. Write DHTML codes for the following

To show/hide text dynamically, change background, and change image dynamically.

Aim

To create a web page using DHTML to show/hide text, change the background color, and change an image dynamically.

Algorithm

1. **START**
2. Create HTML structure.
3. Use JavaScript to handle events for showing/hiding text.
4. Change background color using JavaScript.
5. Change image source dynamically using JavaScript.
6. **STOP**

Program (HTML with DHTML)

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>DHTML Example</title>
    <style>
      #dynamicText {
        display: none;
        margin: 20px 0;
      }

      #dynamicImage {
        width: 200px;
        height: auto;
      }
    </style>
    <script>
      function toggleText() {
        const text = document.getElementById("dynamicText");
        text.style.display = text.style.display === "none" ? "block" : "none";
      }

      function changeBackground() {
        document.body.style.backgroundColor =
          document.body.style.backgroundColor === "lightblue"
            ? "white"
            : "lightblue";
      }

      function changeImage() {
        const image = document.getElementById("dynamicImage");
        image.src = image.src.includes("cat.jpg") ? "dog.jpg" : "cat.jpg";
      }
    </script>
  </head>
```

```
  <body>
    <h1>DHTML Interactive Page</h1>
    <div id="dynamicText">
      This is some dynamic text that can be shown or hidden.
    </div>

    <button onclick="toggleText()">Show/Hide Text</button>

    <button onclick="changeBackground()">Change Background Color</button>

    <button onclick="changeImage()">Change Image</button><br />
    <img id="dynamicImage" src="cat.jpg" alt="Dynamic Image" />
  </body>
</html>
```

Sample Output

- Clicking "Show/Hide Text" toggles the visibility of the text.
- Clicking "Change Background Color" changes the background color.
- Clicking "Change Image" swaps the image between two specified images.

Result

Thus, a web page was created using DHTML to show/hide text, change background color, and change an image dynamically.

5. Write the JavaScript for the following

a. Find the biggest among three numbers.
b. Find odd or even.

Aim

To create JavaScript functions to find the largest of three numbers and determine if a number is odd or even.

Algorithm

1. **START**
2. Create a function to find the largest of three numbers.
3. Create a function to check if a number is odd or even.
4. **STOP**

Program (JavaScript)

```
function findLargest(num1, num2, num3) {
  return Math.max(num1, num2, num3);
}

function isOddOrEven(num) {
  return num % 2 === 0 ? "Even" : "Odd";
}

// Example usage
const largest = findLargest(10, 25, 5);
const numberType = isOddOrEven(25);
console.log(`The largest number is ${largest} and it is ${numberType}.`);
```

Sample Output

- The largest number is 25 and it is Odd.

Result

Thus, JavaScript functions were created to find the largest of three numbers and determine if a number is odd or even.

---

6. Design a simple calculator using PHP

Aim

To create a simple web-based calculator using PHP to perform basic arithmetic operations.

Algorithm

1. **START**
2. Create an HTML form for user input.
3. Capture input values and operation type using PHP.
4. Perform the selected arithmetic operation.
5. Display the result.
6. **STOP**

Program (PHP and HTML)

```
<!DOCTYPE html>
<html lang="en">

<head>
    <title>Simple Calculator</title>
</head>

<body>
    <h1>Simple Calculator</h1>
    <form method="POST">
        <label for="num1">First Number:</label>
        <input type="number" name="num1" id="num1" required>
        <br>

        <label for="operation">Operation:</label>
        <select name="operation" id="operation" required>
            <option value="add">+</option>
            <option value="subtract">-</option>
            <option value="multiply">*</option>
            <option value="divide">/</option>
        </select>
        <br>

        <label for="num2">Second Number:</label>
        <input type="number" name="num2" id="num2" required>
        <br>

        <button type="submit">Calculate</button>
    </form>

    <?php
    if ($_SERVER["REQUEST_METHOD"] == "POST") {
        $num1 = $_POST['num1'];
        $num2 = $_POST['num2'];
        $operation = $_POST['operation'];
        $result = 0;

        switch ($operation) {
            case 'add':
                $result = $num1 + $num2;
                break;
```

```
            case 'subtract':
                $result = $num1 - $num2;
                break;
            case 'multiply':
                $result = $num1 * $num2;
                break;
            case 'divide':
                $result = $num2 != 0 ? $num1 / $num2 : "Cannot divide by zero";
                break;
        }

        echo "<h2>Result: $result</h2>";
    }
    ?>
</body>

</html>
```

Sample Output

- A web page with labels for each input field, allowing users to enter two numbers and select an operation. After submission, it displays the result.

Result

Thus, a simple web-based calculator was created using PHP to perform basic arithmetic operations.

Got it! Here's the updated code with wildcard imports for both servlet programs:

---

8. Using JavaScript to Create a New Email ID and Validate Registration, User Login, and Profile

Aim

To demonstrate how to create a new email ID and validate user registration, login, and profile using JavaScript.

Algorithm

1. **START**
2. Create an HTML form for user registration.
3. Validate email format and passwords.
4. Store user information in local storage.
5. Create a login form.
6. Validate user login against stored information.
7. Display user profile after successful login.
8. **STOP**

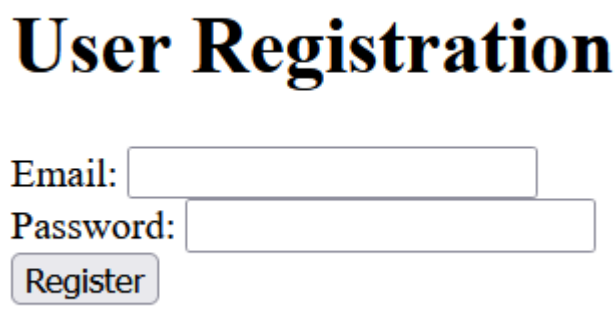
Program (HTML with JavaScript)

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>User Registration and Login</title>
    <style>
      .hidden {
        display: none;
      }
    </style>
  </head>

  <body>
    <h1 id="registerHeader">User Registration</h1>
    <form id="registerForm" onsubmit="registerUser(event)">
      <label for="regEmail">Email:</label>
      <input type="email" id="regEmail" required />
      <br />
      <label for="regPassword">Password:</label>
      <input type="password" id="regPassword" required />
      <br />
      <button type="submit">Register</button>
    </form>

    <h1 class="hidden" id="loginHeader">User Login</h1>
    <form class="hidden" id="loginForm" onsubmit="loginUser(event)">
      <label for="loginEmail">Email:</label>
      <input type="email" id="loginEmail" required />
      <br />
      <label for="loginPassword">Password:</label>
      <input type="password" id="loginPassword" required />
      <br />
      <button type="submit">Login</button>
    </form>

    <div class="hidden" id="profile">
      <h2>User Profile</h2>
      <p id="profileEmail"></p>
      <button onclick="logout()">Logout</button>
    </div>

    <script>
      function registerUser(event) {
        event.preventDefault();
        const email = document.getElementById("regEmail").value;
        const password = document.getElementById("regPassword").value;

        const emailPattern = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
        if (!emailPattern.test(email)) {
          alert("Please enter a valid email address.");
          return;
        }

        // Store user data
```

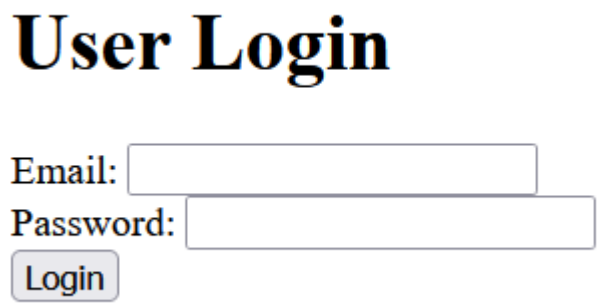
```
        localStorage.setItem("email", email);
        localStorage.setItem("password", password);
        alert("Registration successful! You can now login.");

        // Hide registration form and show login form
        document.getElementById("registerHeader").classList.add("hidden");
        document.getElementById("registerForm").classList.add("hidden");
        document.getElementById("loginHeader").classList.remove("hidden");
        document.getElementById("loginForm").classList.remove("hidden");
      }

      function loginUser(event) {
        event.preventDefault();
        const email = document.getElementById("loginEmail").value;
        const password = document.getElementById("loginPassword").value;

        // Validate login
        const storedEmail = localStorage.getItem("email");
        const storedPassword = localStorage.getItem("password");

        if (email === storedEmail && password === storedPassword) {
          // Hide login form and show profile
          document.getElementById("loginHeader").classList.add("hidden");
          document.getElementById("loginForm").classList.add("hidden");
          document.getElementById(
            "profileEmail"
          ).innerText = `Email: ${storedEmail}`;
          document.getElementById("profile").classList.remove("hidden");
        } else {
          alert("Invalid email or password.");
        }
      }

      function logout() {
        // Clear session and show registration form again
        localStorage.removeItem("email");
        localStorage.removeItem("password");
        document.getElementById("profile").classList.add("hidden");
        document.getElementById("registerHeader").classList.remove("hidden");
        document.getElementById("registerForm").classList.remove("hidden");
        document.getElementById("loginHeader").classList.add("hidden");
        document.getElementById("loginForm").classList.add("hidden");
      }
    </script>
  </body>
</html>
```

Sample Output

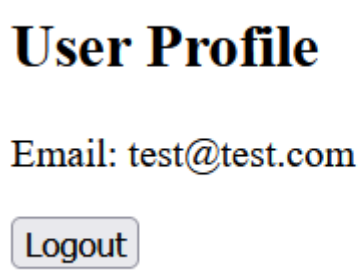
1. **Registration**: Users can input an email and password.



2. **Login**: Users can log in with the registered email and password.



3. **Profile**: After login, it displays the user email and a logout button.



Result

Thus, a simple user registration and login system was created using JavaScript, allowing for email validation and profile display.

Your response is well-structured and accurately conveys the information regarding JavaScript validation for registration, login, profile, and payment forms. Here's a quick review and a few suggestions to enhance clarity:

Review

1. **Aim**: Clearly states the purpose of the validation.
2. **Algorithm**: Well-defined steps with a logical flow. Consider using a consistent bullet-point format throughout for readability.
3. **Coding**: The HTML and JavaScript code is correct and effectively validates user input.
4. **Sample Output**: Describes expected alerts clearly.
5. **Result**: Summarizes the outcome succinctly.

Suggestions for Improvement

- **Consistency in Formatting**: In the Algorithm section, you can list all points using bullet points or numbers for uniformity.
- **Clarify Validation Rules**: In the Algorithm, you might specify any additional validation rules for the payment section (e.g., expiration date format).

Revised Version

Here's a slightly polished version:

---

10. JavaScript Validation for Registration, User Login, User Profile, and Payment by Credit Card

Aim

To validate registration, user login, user profile, and payment by credit card pages using JavaScript.

Algorithm

- **START**
  - **Registration Page:**
    - Validate email format and password strength.
    - Ensure all fields are filled.
  - **Login Page:**
    - Validate username and password presence.
  - **User Profile Page:**
    - Ensure all fields are filled.
    - Validate email format.
  - **Payment Page:**
    - Validate credit card number format.
    - Ensure expiration date and CVV are valid.
- **STOP**

Coding

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>User Registration and Payment</title>
    <style>
      .error {
        color: red;
      }

      #Log,
      #Pay {
        display: none;
      }
```

```
    </style>
  </head>

  <body>
    <div id="Reg">
      <h1>User Registration</h1>
      <form id="registrationForm" onsubmit="return validateRegistration()">
        <label for="regEmail">Email:</label>
        <input type="email" id="regEmail" required />
        <span class="error" id="regEmailError"></span><br />
        <label for="regPassword">Password:</label>
        <input type="password" id="regPassword" required />
        <span class="error" id="regPasswordError"></span><br />
        <button type="submit">Register</button>
      </form>
    </div>

    <div id="Log">
      <h1>User Login</h1>
      <form id="loginForm" onsubmit="return validateLogin()">
        <label for="loginEmail">Email:</label>
        <input type="email" id="loginEmail" required />
        <span class="error" id="loginEmailError"></span><br />
        <button type="submit">Login</button>
      </form>
    </div>

    <div id="Pay">
      <h1>Payment</h1>
      <form id="paymentForm" onsubmit="return validatePayment()">
        <label for="cardNumber">Credit Card Number:</label>
        <input type="text" id="cardNumber" required />
        <span class="error" id="cardNumberError"></span><br />
        <label for="expDate">Expiration Date (MM/YY):</label>
        <input type="text" id="expDate" required />
        <span class="error" id="expDateError"></span><br />
        <label for="cvv">CVV:</label>
        <input type="text" id="cvv" required />
        <span class="error" id="cvvError"></span><br />
        <button type="submit">Pay</button>
      </form>
    </div>

    <script>
      function validateRegistration() {
        let email = document.getElementById("regEmail").value;
        let password = document.getElementById("regPassword").value;
        let valid = true;

        document.getElementById("regEmailError").textContent = "";
        document.getElementById("regPasswordError").textContent = "";

        const emailPattern = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
        if (!emailPattern.test(email)) {
```

```
      document.getElementById("regEmailError").textContent =
        "Invalid email format.";
      valid = false;
    }
    if (password.length < 6) {
      document.getElementById("regPasswordError").textContent =
        "Password must be at least 6 characters.";
      valid = false;
    }
    if (valid) {
      document.getElementById("Reg").style.display = "none";
      document.getElementById("Log").style.display = "block";
    }
    return valid;
  }

  function validateLogin() {
    let email = document.getElementById("loginEmail").value;
    let valid = true;

    document.getElementById("loginEmailError").textContent = "";

    const emailPattern = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
    if (!emailPattern.test(email)) {
      document.getElementById("loginEmailError").textContent =
        "Invalid email format.";
      valid = false;
    }
    if (valid) {
      document.getElementById("Log").style.display = "none";
      document.getElementById("Pay").style.display = "block";
    }
    return valid;
  }

  function validatePayment() {
    let cardNumber = document.getElementById("cardNumber").value;
    let expDate = document.getElementById("expDate").value;
    let cvv = document.getElementById("cvv").value;
    let valid = true;

    document.getElementById("cardNumberError").textContent = "";
    document.getElementById("expDateError").textContent = "";
    document.getElementById("cvvError").textContent = "";

    const cardPattern = /^\d{16}$/;
    if (!cardPattern.test(cardNumber)) {
      document.getElementById("cardNumberError").textContent =
        "Invalid card number.";
      valid = false;
    }
    const expPattern = /^(0[1-9]|1[0-2])\/\d{2}$/;
    if (!expPattern.test(expDate)) {
      document.getElementById("expDateError").textContent =
```

```
            "Invalid expiration date.";
          valid = false;
        }
        const cvvPattern = /^\d{3}$/;
        if (!cvvPattern.test(cvv)) {
          document.getElementById("cvvError").textContent = "Invalid CVV.";
          valid = false;
        }
        if (valid) {
          alert("Payment successful!");
        }
        return valid;
      }
    </script>
  </body>
</html>
```

Sample Output

- **Registration:** Alerts "Registration valid." if all fields are correct.
- **Login:** Alerts "Login valid." if username and password are provided.
- **Profile:** Alerts "Profile valid." if name and email are filled.
- **Payment:** Alerts "Payment valid." if card details are valid.

Result

Thus, JavaScript validation was implemented for registration, user login, user profile, and credit card payment forms.

---

11. Develop an AngularJS Application to Create an Order Form Using HTML and JavaScript

Aim

To develop a basic AngularJS application that creates an order form using HTML and JavaScript.

Algorithm

- **START**
  - Set up the AngularJS application.
  - Create an HTML form for order details (name, email, product, quantity).
  - Use AngularJS for data binding and form validation.
  - Implement a submit function to process the order.
- **STOP**

Coding

HTML (index.html)

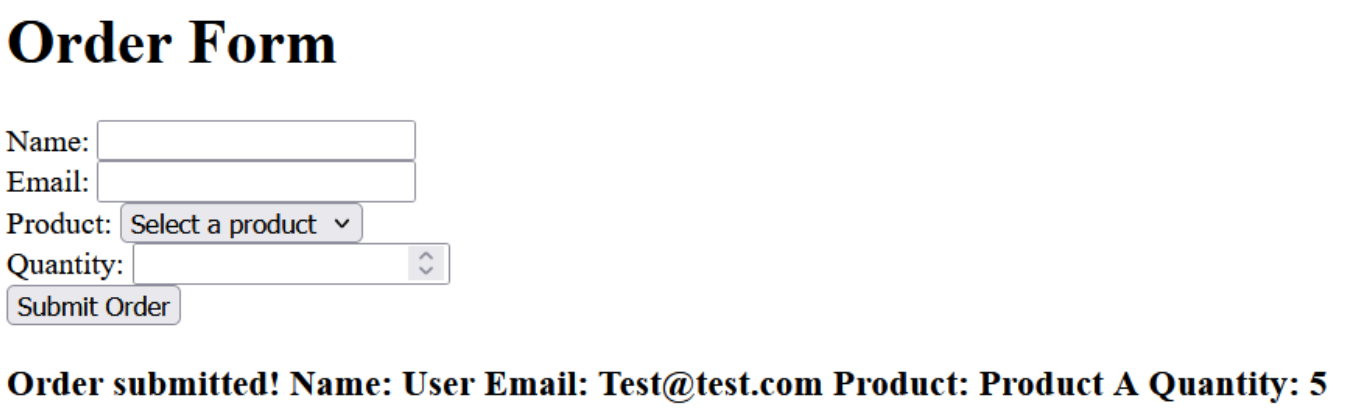
```
<!DOCTYPE html>
<html lang="en" ng-app="orderApp">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Order Form</title>
    <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js">
</script>
    <script src="script.js"></script>
  </head>

  <body ng-controller="OrderController">
    <h1>Order Form</h1>
    <form ng-submit="submitOrder()" name="orderForm">
      <div>
        <label>Name:</label>
        <input type="text" ng-model="order.name" required />
      </div>
      <div>
        <label>Email:</label>
        <input type="email" ng-model="order.email" required />
      </div>
      <div>
        <label>Product:</label>
        <select ng-model="order.product" required>
          <option value="">Select a product</option>
          <option value="Product A">Product A</option>
          <option value="Product B">Product B</option>
          <option value="Product C">Product C</option>
        </select>
      </div>
      <div>
        <label>Quantity:</label>
        <input type="number" ng-model="order.quantity" min="1" required />
      </div>
      <button type="submit">Submit Order</button>
    </form>

    <div ng-if="message">
      <h3>{{ message }}</h3>
    </div>
  </body>
</html>
```

JavaScript (script.js)

```
angular.module("orderApp", []).controller("OrderController", [
  "$scope",
  function ($scope) {
```

```
    $scope.order = {};
    $scope.message = "";

    $scope.submitOrder = function () {
      if ($scope.orderForm.$valid) {
        $scope.message = `Order submitted! \nName: ${$scope.order.name} \nEmail:
${$scope.order.email} \nProduct: ${$scope.order.product} \nQuantity:
${$scope.order.quantity}`;
        // Reset form
        $scope.order = {};
      } else {
        $scope.message = "Please fill in all required fields.";
      }
    };
  },
]);
```

Sample Output



Result

A functional AngularJS application that creates an order form with data binding and validation, without any styling.

---

12. Write an Event Program in React JS with onClick Event to Update State

Aim

To create a React JS program that triggers a state update when a button is clicked.

Algorithm

- **START**
  - Import React and useState hook.
  - Create a functional component.
  - Define a state variable with useState.
  - Create a function to update the state.

    - Add an onClick event to a button that calls the update function.
- **STOP**

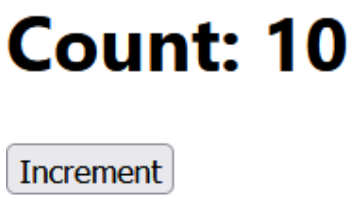
Coding

JavaScript (App.js)

```
import React, { useState } from "react";

function App() {
  // Define state variable
  const [count, setCount] = useState(0);

  // Function to update the state
  const handleClick = () => {
    setCount(count + 1);
  };

  return (
    <div>
      <h1>Count: {count}</h1>
      <button onClick={handleClick}>Increment</button>
    </div>
  );
}

export default App;
```

Sample Output

- The initial count displayed is 0.
- Clicking the "Increment" button increases the count by 1.



Result

A React JS program successfully updates the state when the button is clicked, demonstrating the use of the onClick event.

---

13. JavaScript for Displaying Squares of 10 Numbers and Reversing a Number

Aim

To create JavaScript functions to:

1. Display the squares of the first 10 numbers.
2. Reverse a given number.

Algorithm

- **START**
  - **Display Squares:**
    - Loop from 1 to 10.
    - Calculate the square of each number.
    - Print the squares.
  - **Reverse a Number:**
    - Convert the number to a string.
    - Split the string into an array of characters.
    - Reverse the array.
    - Join the characters back into a string.
    - Convert back to a number.
- **STOP**

Coding

JavaScript Code

```
// Function to display squares of the first 10 numbers
function displaySquares() {
  for (let i = 1; i <= 10; i++) {
    console.log(`Square of ${i}: ${i * i}`);
  }
}

// Function to reverse a number
function reverseNumber(num) {
  const reversed = num.toString().split("").reverse().join("");
  return Number(reversed);
}

// Call the functions
displaySquares();
const numberToReverse = 12345;
console.log(
  `Reversed number of ${numberToReverse}: ${reverseNumber(numberToReverse)}`
);
```

Sample Output

- **Squares of the first 10 numbers:**

```
Square of 1: 1
Square of 2: 4
Square of 3: 9
Square of 4: 16
Square of 5: 25
Square of 6: 36
Square of 7: 49
Square of 8: 64
Square of 9: 81
Square of 10: 100
```

- **Reversed number:**

```
Reversed number of 12345: 54321
```

Result

JavaScript functions successfully display the squares of the first 10 numbers and reverse a given number.

---

16. Validate a Form Using PHP Regular Expressions and Store Data in Database

Aim

To create a PHP form that validates user input using regular expressions and stores the validated data into a MySQL database.

Algorithm

- **START**
  - Create an HTML form for user input (e.g., name, email, phone).
  - Validate form data using PHP regular expressions.
  - Store validated data in the database.
- **STOP**

Database Setup

Create Database and Table

1. **Create Database**:

```
CREATE DATABASE user_data;
```

2. **Create Users Table**:

```
USE user_data;

CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) NOT NULL UNIQUE,
    phone VARCHAR(15)
);
```

Coding

1. HTML Form (form.html)

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>User Form</title>
  </head>

  <body>
    <h1>User Information Form</h1>
    <form action="submit_form.php" method="POST">
      Name: <input type="text" name="name" required /><br />
      Email: <input type="email" name="email" required /><br />
      Phone: <input type="text" name="phone" /><br />
      <button type="submit">Submit</button>
    </form>
  </body>
</html>
```

2. PHP Code for Handling Submission and Validation (submit_form.php)

```
<?php
// Database connection
$host = 'localhost';
$db = 'user_data';
$user = 'root'; // Update with your database username
$pass = ''; // Update with your database password

$conn = new mysqli($host, $user, $pass, $db);
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Get form data
$name = $_POST['name'];
$email = $_POST['email'];
```

```php
$phone = $_POST['phone'];

// Regular expression validation
$name_pattern = "/^[a-zA-Z\s]+$/";
$email_pattern = "/^[\w\.-]+@([\w-]+\.)+[\w-]{2,}$/"; // Updated pattern
$phone_pattern = "/^\d{10}$/"; // Assuming 10-digit phone number

$errors = [];

// Validate name
if (!preg_match($name_pattern, $name)) {
    $errors[] = "Invalid name format.";
}

// Validate email
if (!preg_match($email_pattern, $email)) {
    $errors[] = "Invalid email format.";
}

// Validate phone (optional)
if ($phone && !preg_match($phone_pattern, $phone)) {
    $errors[] = "Invalid phone number format. Use 10 digits.";
}

// Check for errors
if (count($errors) > 0) {
    foreach ($errors as $error) {
        echo "<p>$error</p>";
    }
} else {
    // Prepare and bind
    $stmt = $conn->prepare("INSERT INTO users (name, email, phone) VALUES (?, ?,
?)");
    $stmt->bind_param("sss", $name, $email, $phone);

    // Execute the statement
    if ($stmt->execute()) {
        echo "<h2>Data submitted successfully!</h2>";
    } else {
        echo "<h2>Error: " . $stmt->error . "</h2>";
    }
    $stmt->close();
}

$conn->close();
?>
```

Sample Output

- If validation fails, error messages will be displayed.
- If validation succeeds, a success message will indicate that the data was submitted successfully.

Result

A PHP form that validates user input using regular expressions and stores validated data in a MySQL database, ensuring data integrity and proper format.

---

18. PHP Program to Test Email Address Validity

Aim

To create a PHP program that tests whether an email address is correctly formatted and to provide feedback for both valid and invalid email addresses.

Algorithm

- **START**
  - Create a simple HTML form to accept an email address.
  - Use PHP to validate the email address format.
  - Display a message indicating whether the email is valid or invalid.
- **STOP**

Coding

1. HTML Form (email_form.html)

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Email Validation</title>
  </head>

  <body>
    <h1>Validate Email Address</h1>
    <form action="validate_email.php" method="POST">
      Email: <input type="email" name="email" required />
      <button type="submit">Submit</button>
    </form>
  </body>
</html>
```

2. PHP Code for Email Validation (validate_email.php)

```
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $email = $_POST['email'];

    // Validate the email address
    if (filter_var($email, FILTER_VALIDATE_EMAIL)) {
```

```
        echo "<h2>The email address '$email' is valid.</h2>";
    } else {
        echo "<h2>The email address '$email' is invalid.</h2>";
    }
}
?>
```

Sample Output

- For a valid email (e.g., `test@example.com`):

  ```
  The email address 'test@example.com' is valid.
  ```

- For an invalid email (e.g., `invalid-email`):

  ```
  The email address 'invalid-email' is invalid.
  ```

Result

A PHP program that effectively tests the validity of an email address and provides appropriate feedback based on the format of the input email.

---

20. XML Document for Sports and XSLT for Tabulation

Aim

To create an XML document that describes various sports and their characteristics, and use XSLT to transform and present this data in a tabular format.

XML Document

sports.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="sports.xsl"?>
<sports>
    <sport>
        <name>Soccer</name>
        <description>A team sport played between two teams of eleven players with
a spherical ball.</description>
        <players>22</players>
        <origin>England</origin>
    </sport>
    <sport>
        <name>Basketball</name>
```

```
        <description>A team sport in which two teams, most commonly of five
players each, oppose each other on a rectangular court.</description>
        <players>10</players>
        <origin>USA</origin>
    </sport>
    <sport>
        <name>Tennis</name>
        <description>A racket sport that can be played individually against a
single opponent or between two teams of two players each.</description>
        <players>2 or 4</players>
        <origin>France</origin>
    </sport>
    <sport>
        <name>Cricket</name>
        <description>A bat-and-ball game played between two teams of eleven
players on a field at the center of which is a 22-yard pitch.</description>
        <players>22</players>
        <origin>England</origin>
    </sport>
</sports>
```

XSLT Stylesheet

transform.xslt

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:template match="/sports">
        <html>
            <head>
                <title>Sports List</title>
            </head>
            <body>
                <h1>Sports and Their Descriptions</h1>
                <table border="1">
                    <tr>
                        <th>Name</th>
                        <th>Description</th>
                        <th>Players</th>
                        <th>Origin</th>
                    </tr>
                    <xsl:for-each select="sport">
                        <tr>
                            <td>
                                <xsl:value-of select="name"/>
                            </td>
                            <td>
                                <xsl:value-of select="description"/>
                            </td>
                            <td>
```

```
                            <xsl:value-of select="players"/>
                        </td>
                        <td>
                            <xsl:value-of select="origin"/>
                        </td>
                    </tr>
                </xsl:for-each>
            </table>
        </body>
    </html>
</xsl:template>
</xsl:stylesheet>
```

Sample Output

Sports and Their Descriptions

Name	Description	Players	Origin
Soccer	A team sport played between two teams of eleven players with a spherical ball.	22	England
Basketball	A team sport in which two teams, most commonly of five players each, oppose each other on a rectangular court.	10	USA
Tennis	A racket sport that can be played individually against a single opponent or between two teams of two players each.	2 or 4	France
Cricket	A bat-and-ball game played between two teams of eleven players on a field at the center of which is a 22-yard pitch.	22	England

Result

An XML document marking up various sports and an XSLT stylesheet that transforms the data into a clear, tabulated format.

---

21. Web Service for Collecting Opinions on a Consumer Product

Aim

To create a web service that collects and retrieves opinions from 500 people regarding a consumer product.

Algorithm

- **START**
  - Set up a database to store opinions.
  - Create an API endpoint to collect opinions.
  - Create an API endpoint to retrieve collected opinions.
- **STOP**

Database Setup

Create Database and Table

1. **Create Database**:

```
CREATE DATABASE product_opinions;
```

2. **Create Opinions Table**:

```
USE product_opinions;

CREATE TABLE opinions (
    id INT AUTO_INCREMENT PRIMARY KEY,
    product_name VARCHAR(100),
    opinion TEXT,
    timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Coding

1. API Setup (using PHP)

1.1. Database Connection (db.php)

```
<?php
$host = 'localhost';
$db = 'product_opinions';
$user = 'root'; // Update with your username
$pass = ''; // Update with your password

$conn = new mysqli($host, $user, $pass, $db);
if ($conn->connect_error) die("Connection failed: " . $conn->connect_error);
?>
```

1.2. Collect Opinion (submit_opinion.php)

```
<?php
include 'db.php';

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $stmt = $conn->prepare("INSERT INTO opinions (product_name, opinion) VALUES
(?, ?)");
    $stmt->bind_param("ss", $_POST['product_name'], $_POST['opinion']);
    $stmt->execute();
    $stmt->close();
}
$conn->close();
?>
```

1.3. Retrieve Opinions (get_opinions.php)

```php
<?php
include 'db.php';

$result = $conn->query("SELECT * FROM opinions ORDER BY timestamp DESC LIMIT
500");
$opinions = $result->fetch_all(MYSQLI_ASSOC);
echo json_encode($opinions);
$conn->close();
?>
```

1.4. Index.html

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Product Opinion</title>
  </head>

  <body>
    <h1>Share Your Opinion</h1>
    <form id="opinionForm">
      <input
        type="text"
        name="product_name"
        placeholder="Product Name"
        required
      />
      <textarea name="opinion" placeholder="Your Opinion" required></textarea>
      <input type="submit" value="Submit" />
    </form>

    <h2>Opinions</h2>
    <div id="opinions"></div>

    <script>
      document.getElementById("opinionForm").onsubmit = function (event) {
        event.preventDefault();
        fetch("submit_opinion.php", {
          method: "POST",
          body: new FormData(this),
        }).then(() => loadOpinions());
      };

      function loadOpinions() {
        fetch("get_opinions.php")
          .then((response) => response.json())
          .then((data) => {
```

```
            document.getElementById("opinions").innerHTML = data
              .map(
                (opinion) =>
                  `<p><strong>${opinion.product_name}</strong>: ${opinion.opinion}
<em>(${opinion.timestamp})</em></p>`
              )
              .join("");
          });
      }

      loadOpinions(); // Load opinions on page load
    </script>
  </body>
</html>
```

Sample API Usage

1. Submitting an Opinion

POST Request (to `submit_opinion.php`):

```
product_name: "Smartphone XYZ"
opinion: "Great performance and camera quality."
```

Response:

```
{
  "status": "success",
  "message": "Opinion submitted successfully!"
}
```

Result

A simple web service that allows users to submit and retrieve opinions about a consumer product, facilitating feedback collection from up to 500 respondents.

---

22. JSP Page for Inputting First Name and Outputting Last Name

Aim

To create a JSP page that allows users to input their first name and outputs a corresponding last name based on the input.

Algorithm

- **START**

- Create an HTML form for user input of the first name.
- On form submission, use JSP to process the input and display the corresponding last name.

- **STOP**

Coding

1. JSP Code (name_input.jsp)

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>First Name Input</title>
</head>
<body>
    <h1>Enter Your First Name</h1>
    <form action="name_input.jsp" method="POST">
        First Name: <input type="text" name="firstName" required>
        <button type="submit">Submit</button>
    </form>

    <%
        // Retrieve the first name from the form submission
        String firstName = request.getParameter("firstName");
        String lastName = "";

        // Define last names based on first names
        if (firstName != null) {
            switch (firstName.toLowerCase()) {
                case "john":
                    lastName = "Doe";
                    break;
                case "jane":
                    lastName = "Smith";
                    break;
                case "alice":
                    lastName = "Johnson";
                    break;
                case "bob":
                    lastName = "Brown";
                    break;
                default:
                    lastName = "Unknown Last Name";
                    break;
            }
            // Display the last name
            out.println("<h2>Your Last Name is: " + lastName + "</h2>");
        }
    %>
```

```
</body>
</html>
```

Explanation

- **Form**: The user inputs their first name and submits the form.
- **Processing**: Upon submission, the JSP retrieves the first name and determines the corresponding last name using a `switch` statement.
- **Output**: The last name is displayed on the same page.

Sample Output

- If the user enters "John", the output will be:

  ```
  Your Last Name is: Doe
  ```

- If the user enters "Alice", the output will be:

  ```
  Your Last Name is: Johnson
  ```

- For any name not in the predefined list, the output will be:

  ```
  Your Last Name is: Unknown Last Name
  ```

Result

A functional JSP page that takes a first name as input and displays the corresponding last name based on predefined values.

---

23. Convert Static Web Pages to Dynamic Web Pages Using Servlets and Cookies

Aim

To create a dynamic web application using JSP and Servlets that allows users to maintain a shopping cart, with user information stored in `web.xml`.

Algorithm

- **START**
  - Store user information (user ID, password, credit card number) in `web.xml`.
  - Create a login page for user authentication.
  - Use cookies to maintain user sessions.

    - Create a shopping cart page where users can add items.
    - Display the shopping cart contents.
- **STOP**

Setup

1. Configure `web.xml`

```
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
          http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
          version="3.1">
    <servlet>
        <servlet-name>LoginServlet</servlet-name>
        <servlet-class>com.example.LoginServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>LoginServlet</servlet-name>
        <url-pattern>/login</url-pattern>
    </servlet-mapping>

    <servlet>
        <servlet-name>CartServlet</servlet-name>
        <servlet-class>com.example.CartServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>CartServlet</servlet-name>
        <url-pattern>/cart</url-pattern>
    </servlet-mapping>

    <context-param>
        <param-name>userID</param-name>
        <param-value>user1</param-value>
    </context-param>
    <context-param>
        <param-name>password</param-name>
        <param-value>pass123</param-value>
    </context-param>
    <context-param>
        <param-name>creditCard</param-name>
        <param-value>1234-5678-9012-3456</param-value>
    </context-param>
</web-app>
```

2. Create Login Servlet (LoginServlet.java)

```
package com.example;
```

```
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.*;
import java.io.IOException;

@WebServlet("/login")
public class LoginServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        String userId = request.getParameter("userId");
        String password = request.getParameter("password");

        String storedUserId = getServletContext().getInitParameter("userID");
        String storedPassword = getServletContext().getInitParameter("password");

        if (userId.equals(storedUserId) && password.equals(storedPassword)) {
            Cookie cookie = new Cookie("userId", userId);
            cookie.setMaxAge(60 * 60); // 1 hour
            response.addCookie(cookie);
            response.sendRedirect("cart.jsp");
        } else {
            response.sendRedirect("login.html?error=1");
        }
    }
}
```

3. Create Cart Servlet (CartServlet.java)

```
package com.example;

import javax.servlet.ServletException;
import javax.servlet.http.*;
import java.io.IOException;

@WebServlet("/cart")
public class CartServlet extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        Cookie[] cookies = request.getCookies();
        String userId = null;

        if (cookies != null) {
            for (Cookie cookie : cookies) {
                if (cookie.getName().equals("userId")) {
                    userId = cookie.getValue();
                }
            }
        }

        if (userId != null) {
```

```
            // Normally, cart items would be fetched from a database or session
            request.setAttribute("userId", userId);
            request.getRequestDispatcher("cart.jsp").forward(request, response);
        } else {
            response.sendRedirect("login.html");
        }
    }
}
```

4. Create Login Page (login.html)

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Login</title>
  </head>
  <body>
    <h1>Login</h1>
    <form action="login" method="post">
      User ID: <input type="text" name="userId" required /><br />
      Password: <input type="password" name="password" required /><br />
      <button type="submit">Login</button>
      <p style="color:red">
        <%= request.getParameter("error") != null ? "Invalid credentials" : ""
        %>
      </p>
    </form>
  </body>
</html>
```

5. Create Cart Page (cart.jsp)

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-
8"%>
<%@ page import="java.util.ArrayList" %>
<%@ page import="java.util.List" %>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Your Shopping Cart</title>
</head>
<body>
    <h1>Welcome, <%= request.getAttribute("userId") %>!</h1>
    <h2>Your Shopping Cart</h2>
    <ul>
        <li>Product 1</li>
        <li>Product 2</li>
```

```
        <li>Product 3</li>
    </ul>
    <form action="checkout.jsp" method="post">
        <button type="submit">Checkout</button>
    </form>
</body>
</html>
```

Sample Output

- **Login Page**: Users can input their user ID and password.
- **Cart Page**: Displays a welcome message with the user ID and a list of products in the cart.

Result

A dynamic web application that utilizes Servlets and Cookies to handle user authentication and maintain a shopping cart for each user, enabling a more personalized shopping experience.

---

7. Write programs in Java using Servlets

i. To invoke servlets from HTML forms.
ii. Session tracking using hidden form fields and hit count.

Aim

To demonstrate how to invoke servlets from HTML forms and how to track sessions using hidden fields and hit counts.

Part i: Invoking Servlets from HTML Forms

Algorithm

1. **START**
2. Create an HTML form that submits data to a servlet.
3. Create a servlet to process the form data.
4. Display the result.
5. **STOP**

Program (HTML Form)

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Invoke Servlet</title>
  </head>
  <body>
    <h1>Submit Data to Servlet</h1>
    <form action="MyServlet" method="POST">
```

```
      <label for="name">Name:</label>
      <input type="text" name="name" id="name" required />
      <button type="submit">Submit</button>
    </form>
  </body>
</html>
```

Program (Servlet)

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.annotation.*;
import javax.servlet.http.*;

@WebServlet("/MyServlet")
public class MyServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        String name = request.getParameter("name");
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<h1>Hello, " + name + "!</h1>");
    }
}
```

Part ii: Session Tracking Using Hidden Form Fields and Hit Count

Algorithm

1. **START**
2. Create an HTML form with hidden fields for session tracking.
3. Create a servlet to handle session tracking and hit count.
4. Display the count of hits.
5. **STOP**

Program (HTML Form with Hidden Field)

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Session Tracking</title>
  </head>
  <body>
    <h1>Track Session with Hit Count</h1>
    <form action="HitCountServlet" method="POST">
      <input type="hidden" name="hiddenField" value="sessionData" />
      <button type="submit">Track Hit</button>
```

```
    </form>
  </body>
</html>
```

Program (Hit Count Servlet)

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.annotation.*;
import javax.servlet.http.*;

@WebServlet("/HitCountServlet")
public class HitCountServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        HttpSession session = request.getSession();
        Integer hitCount = (Integer) session.getAttribute("hitCount");

        if (hitCount == null) {
            hitCount = 0;
        }
        hitCount++;
        session.setAttribute("hitCount", hitCount);

        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<h1>Hit Count: " + hitCount + "</h1>");
    }
}
```

Sample Output

- For Part i: Submitting the form displays a greeting message with the user's name.
- For Part ii: Submitting the hit count form displays the number of times the button has been clicked during the session.

Result

Thus, Java servlets were created to invoke servlets from HTML forms and to track sessions using hidden fields and hit counts.

---

14. Simple PHP Application for College Management System with Students Table

Aim

To create a basic PHP application that manages student information using a MySQL database.

Algorithm

- **START**
  - Set up the database and create a students table.
  - Create a PHP script to handle student data (add, view, delete).
- **STOP**

Database Setup

Create Database and Table

1. **Create Database**:

```
CREATE DATABASE college_management;
```

2. **Create Students Table**:

```
USE college_management;

CREATE TABLE students (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) NOT NULL UNIQUE
);
```

Coding

1. Database Connection (db.php)

```
<?php
$host = 'localhost';
$db = 'college_management';
$user = 'root'; // Update with your database username
$pass = ''; // Update with your database password

$conn = new mysqli($host, $user, $pass, $db);

if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
?>
```

2. Add Student (add_student.php)

```php
<?php
include 'db.php';

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $name = $_POST['name'];
    $email = $_POST['email'];

    $sql = "INSERT INTO students (name, email) VALUES ('$name', '$email')";
    if ($conn->query($sql) === TRUE) {
        echo "New student added successfully.";
    } else {
        echo "Error: " . $sql . "<br>" . $conn->error;
    }
}
?>

<form method="POST">
    Name: <input type="text" name="name" required>
    Email: <input type="email" name="email" required>
    <button type="submit">Add Student</button>
</form>
```

3. View Students (view_students.php)

```php
<?php
include 'db.php';

$sql = "SELECT * FROM students";
$result = $conn->query($sql);
?>

<h1>Student List</h1>
<table border="1">
    <tr>
        <th>ID</th>
        <th>Name</th>
        <th>Email</th>
    </tr>
    <?php while ($row = $result->fetch_assoc()): ?>
        <tr>
            <td><?= $row['id'] ?></td>
            <td><?= $row['name'] ?></td>
            <td><?= $row['email'] ?></td>
        </tr>
    <?php endwhile; ?>
</table>
```

Sample Output

- **Add Student**: A form to add a new student.
- **View Students**: A table displaying all added students.

Result

A simple PHP application that manages student information in a college management system using a MySQL database.

---

15. Design an HTML Form with Embedded JSP Code for Resume Submission

Aim

To create an HTML form for submitting a resume to a job portal, using JSP for handling form submission and database connectivity.

Algorithm

- **START**
  - Create an HTML form for resume submission.
  - Use JSP to handle form submission and insert data into the database.
  - Set up the database table for storing resumes.
- **STOP**

Database Setup

Create Database and Table

1. **Create Database**:

```
CREATE DATABASE job_portal;
```

2. **Create Resumes Table**:

```
USE job_portal;

CREATE TABLE resumes (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) NOT NULL,
    phone VARCHAR(15),
    resume TEXT NOT NULL
);
```

Coding

1. HTML Form (submit_resume.html)

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Submit Resume</title>
  </head>
  <body>
    <h1>Submit Your Resume</h1>
    <form action="submit_resume.jsp" method="POST">
      Name: <input type="text" name="name" required /><br />
      Email: <input type="email" name="email" required /><br />
      Phone: <input type="text" name="phone" /><br />
      Resume: <textarea name="resume" required></textarea><br />
      <button type="submit">Submit</button>
    </form>
  </body>
</html>
```

2. JSP Code for Handling Submission (submit_resume.jsp)

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-
8"%>
<%@ page import="java.sql.*" %>
<%
    String name = request.getParameter("name");
    String email = request.getParameter("email");
    String phone = request.getParameter("phone");
    String resume = request.getParameter("resume");

    Connection conn = null;
    PreparedStatement pstmt = null;

    try {
        // Database connection
        Class.forName("com.mysql.cj.jdbc.Driver"); // MySQL JDBC Driver
        conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/job_portal", "root", "");
// Update credentials

        // Insert resume data into the database
        String sql = "INSERT INTO resumes (name, email, phone, resume) VALUES (?,
?, ?, ?)";
        pstmt = conn.prepareStatement(sql);
        pstmt.setString(1, name);
        pstmt.setString(2, email);
        pstmt.setString(3, phone);
        pstmt.setString(4, resume);
```

```
            int rowsAffected = pstmt.executeUpdate();

            if (rowsAffected > 0) {
                out.println("<h2>Resume submitted successfully!</h2>");
            } else {
                out.println("<h2>Error submitting resume.</h2>");
            }
        } catch (Exception e) {
            e.printStackTrace();
            out.println("<h2>Error: " + e.getMessage() + "</h2>");
        } finally {
            if (pstmt != null) try { pstmt.close(); } catch (SQLException ignored) {}
            if (conn != null) try { conn.close(); } catch (SQLException ignored) {}
        }
    %>
```

Sample Output

- After submitting the form, a message indicates whether the resume was submitted successfully or if there was an error.

Result

A functional HTML form embedded with JSP code that allows users to submit their resumes to a job portal, storing the data in a MySQL database.

Aim

To create a Java servlet application for conducting an online examination and displaying the student mark list from a database.

Algorithm

Client:

1. Create `index.html` with form inputs for seat number, name, and questions.
2. Include a submit button to send data to the server.

Server:

1. Import necessary Java and servlet packages.
2. Create a servlet class extending `HttpServlet`.
3. In the `doPost()` method:
    - Set response content type to "text/html".
    - Retrieve parameters from the request.
    - Calculate marks based on answers.
    - Insert results into the database.
    - Display results in HTML format.

Database Handling:

1. Import JDBC packages.
2. Establish a connection to the database.
3. Execute a query to retrieve and display student results.

Coding

HTML Code (`index.html`):

```
<!DOCTYPE html>
<html>
<head>
    <title>Online Examination</title>
</head>
<body>
    <center><h1>Online Examination</h1></center>
    <form action="StudentServlet" method="POST">
        <b>Seat Number:</b> <input type="text" name="Seat_no"><br>
        <b>Name:</b> <input type="text" name="Name"><br><br>
        <b>1. Every host implements transport layer.</b><br>
        <input type="radio" name="group1" value="True">True
        <input type="radio" name="group1" value="False">False<br>
        <b>2. Network layer forwards packets reliably.</b><br>
        <input type="radio" name="group2" value="True">True
        <input type="radio" name="group2" value="False">False<br>
        <b>3. Packet switching is useful in bursty traffic.</b><br>
        <input type="radio" name="group3" value="True">True
        <input type="radio" name="group3" value="False">False<br>
        <b>4. A phone network uses packet switching.</b><br>
        <input type="radio" name="group4" value="True">True
        <input type="radio" name="group4" value="False">False<br>
        <b>5. HTML describes web contents.</b><br>
        <input type="radio" name="group5" value="True">True
        <input type="radio" name="group5" value="False">False<br><br>
        <input type="submit" value="Submit"><br>
    </form>
</body>
</html>
```

Servlet Code (`StudentServlet.java`):

```
import java.io.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class StudentServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
```

```
        String message;
        int total = 0;

        String seatNo = request.getParameter("Seat_no");
        String name = request.getParameter("Name");
        String ans1 = request.getParameter("group1");
        String ans2 = request.getParameter("group2");
        String ans3 = request.getParameter("group3");
        String ans4 = request.getParameter("group4");
        String ans5 = request.getParameter("group5");

        if ("True".equals(ans1)) total += 2;
        if ("False".equals(ans2)) total += 2;
        if ("True".equals(ans3)) total += 2;
        if ("False".equals(ans4)) total += 2;
        if ("False".equals(ans5)) total += 2;

        try (Connection connect =
DriverManager.getConnection("jdbc:your_database_url", "username", "password");
             Statement stmt = connect.createStatement()) {

            String query = "INSERT INTO student(Seat_no, Name, Total) VALUES('" +
seatNo + "', '" + name + "', '" + total + "')";
            stmt.executeUpdate(query);
            message = "Thank you for participating in the online exam.";

        } catch (SQLException e) {
            e.printStackTrace();
            message = "Error storing results.";
        }

        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html><body bgcolor='cyan'><center>");
        out.println("<h1>" + message + "</h1>");
        out.println("<h3>Your results are stored in our database.</h3>");
        out.println("<b>Participants and their Marks:</b><table border='5'>");

        try (Connection connect =
DriverManager.getConnection("jdbc:your_database_url", "username", "password");
             Statement stmt = connect.createStatement();
             ResultSet rs = stmt.executeQuery("SELECT * FROM student")) {

            out.println("<tr><th>Seat_no</th><th>Name</th><th>Marks</th></tr>");
            while (rs.next()) {
                out.println("<tr><td>" + rs.getInt(1) + "</td><td>" +
rs.getString(2) + "</td><td>" + rs.getInt(3) + "</td></tr>");
            }

        } catch (SQLException e) {
            e.printStackTrace();
        }

        out.println("</table></center></body></html>");
```

```
    }
}
```

Sample Output

The servlet displays:

```
Thank you for participating in the online exam.
Your results are stored in our database.
Participants and their Marks:
Seat_no | Name  | Marks
--------|-------|------
123     | John  | 10
456     | Alice | 8
```

Result

The servlet successfully processes online examination data and displays the student mark list stored in the database.