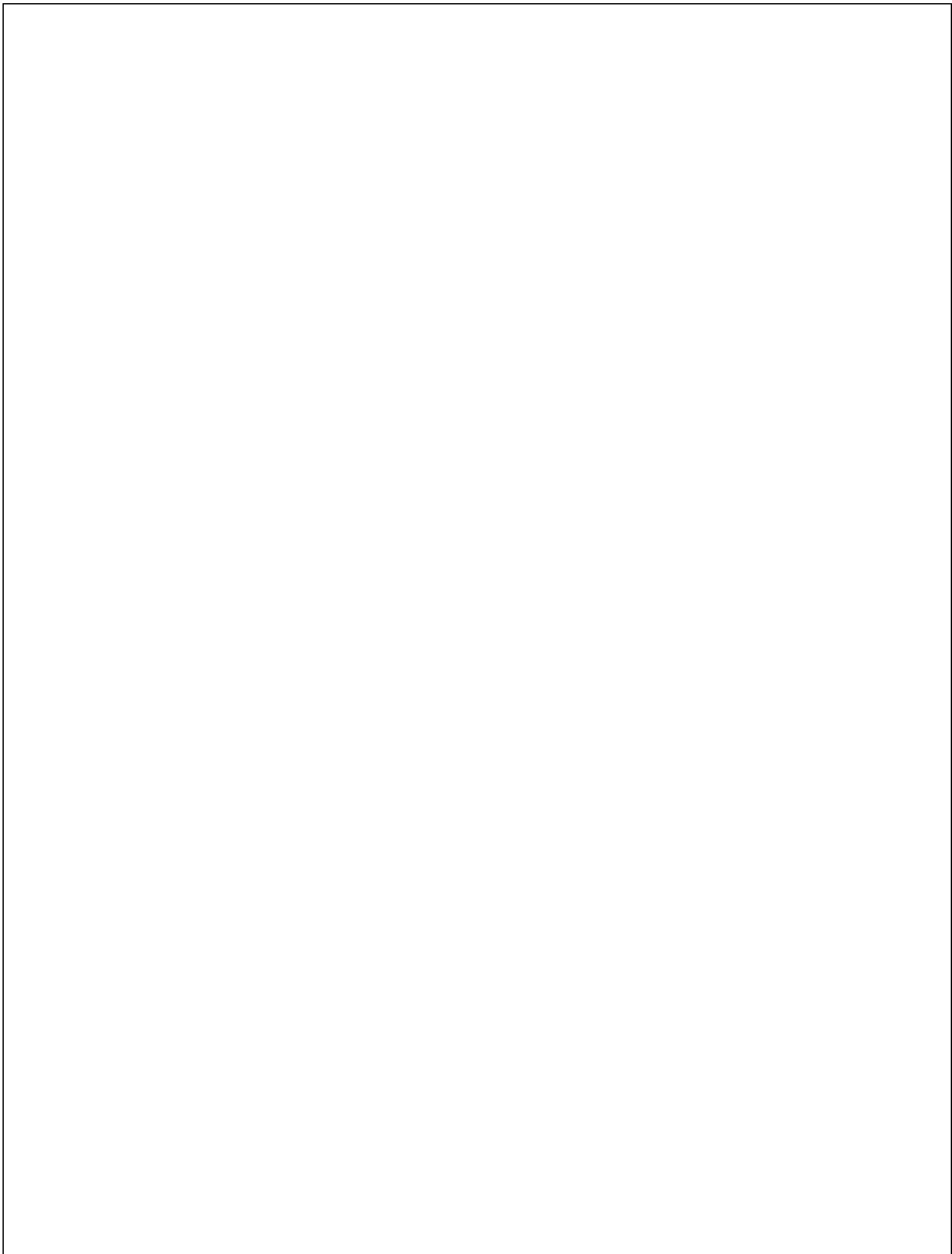


INDEX:-



EX NO: 01

DATE:

WRITE AN HTML CODE TO DISPLAY A LIST OF “BEST FEATURES OF THE INTERNET” USING ORDERED AND UNORDERED LISTS.

AIM:

To create an HTML web page that displays the best features of the internet using ordered and unordered lists, including sub-lists with different numbering lists.

ALGORITHM:

- **STEP 1:** Start the program.
- **STEP 2:** Open the HTML document with <!DOCTYPE html> tag.
- **STEP 3:** Add <html> and <head> sections.
- **STEP 4:** Inside <head>, give the page a title using <title>.
- **STEP 5:** In the <body> section:-
 1. Add a heading <h1> for the page title.
 2. Create a unordered list for main features.
 3. Insert list items for each features.
 4. For sub-features, use needed nested or tags with type attributes (a,1,etc...).
- **STEP 6:** Close all opened tags properly (, , </body>, </html>).
- **STEP 7:** Stop the program.

PROGRAM:

```
<!DOCTYPE html>

<html>
  <head>
    <title>The Best Features of the Internet</title>
  </head>
  <body>
    <h1>The Best Features of the Internet</h1>
    <ul>
      <li>You meet new people from countries around the world.</li>
```

OUTPUT:

```
<li>You have access to new media as it becomes public.</li>
<ul type="a">
    <li>New games</li>
    <li>New applications</li>
</ul>

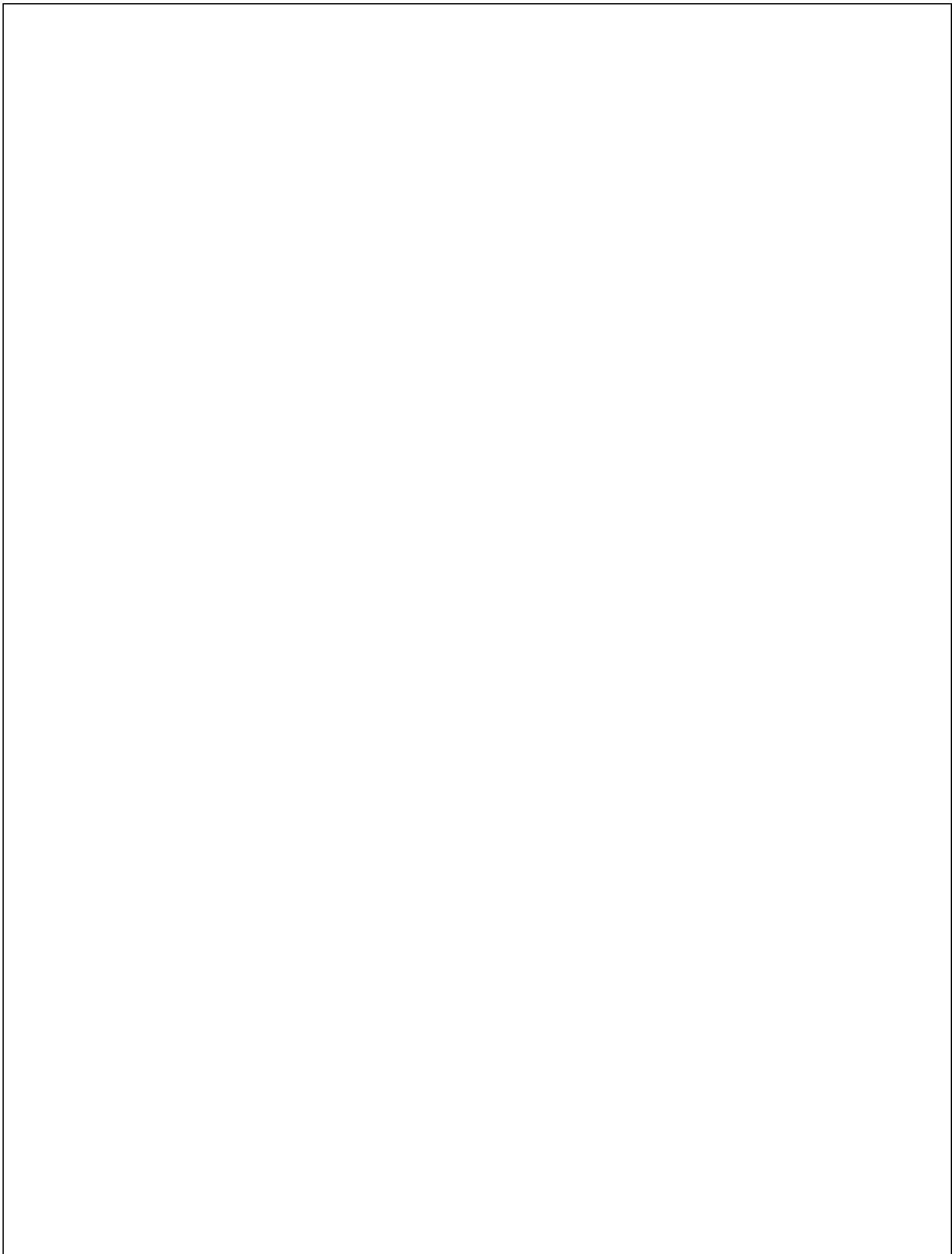
<ol type="1">
    <li>For business</li>
    <li>For pleasure</li>
</ol>

<li>Around the clock news</li>
<li>Search engines</li>
<li>Shopping</li>
<li>Programming</li>

<ol type="1">
    <li>XML</li>
    <li>Java</li>
    <li>HTML</li>
    <li>JavaScript</li>
    <li>New language</li>
</ol>

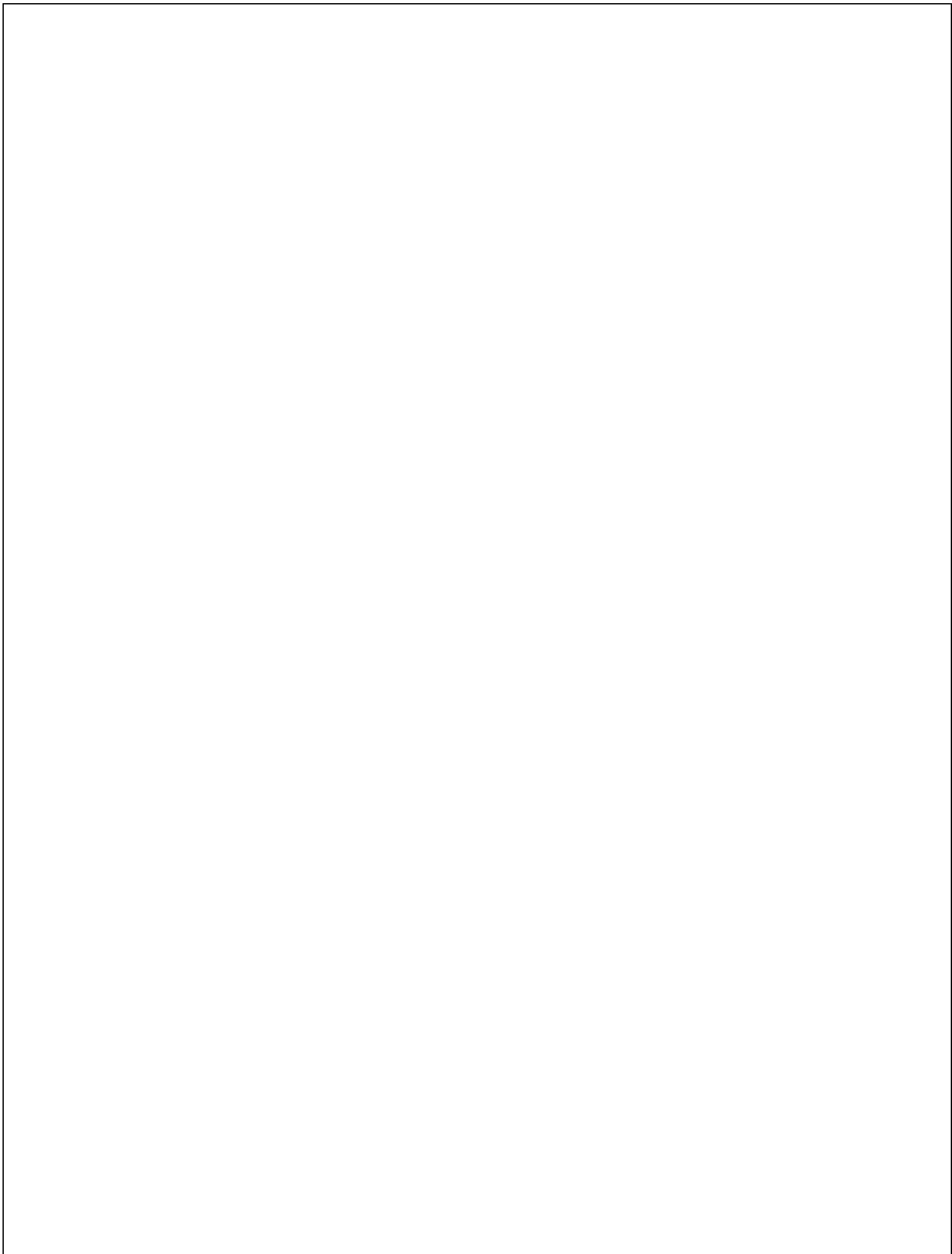
<li>Links</li>
<li>Keeping in touch with old friends</li>
<li>It's the technology of the future!</li>
</ul>

</body></html>
```



RESULT:

Thus the above program Best features of the internet using ordered and unordered lists has been implemented and verified successfully.



Ex No: 02

Date:

WRITE AN HTML CODE TO DISPLAY A CLASS TIME TABLE USING THE <TABLE> TAG.

AIM:

To design a simple HTML web page that displays the class timetable in a tabular Format using the <table> tag with rows, columns and headings.

ALGORITHM:

- **STEP 1:** Start the program.
- **STEP 2:** Begin the HTML document with <!DOCTYPE html> and <html> tags.
- **STEP 3:** Add <head> section with <title> and optional CSS styling for the table.
- **STEP 4:** In the <body> section:
 1. Insert a heading <h2> for the timetable title.
 2. Create a <table> and define headers <th> for days and time slots.
 3. Use <tr> for each row(day) and <td> for each timetable entry.
 4. Merge cells where required using the row span and Col span attribute.
- **STEP 5:** Close the table and HTML tags.
- **STEP 6:** Stop the program.

PROGRAM:

```
<!DOCTYPE html>

<html>

<head>

    <title> Class Timetable – CSE SEM III/V </title>

    <style>

        Table {

            Border-collapse: collapse;

            Width:100%;

            Text-align: centre;

        }

    </style>

```

OUTPUT:

```

th, td {
    border:1px solid black;
    padding:6px;
}

th {
    background-color: #f2f2f2;
}

</style>

</head>

<body>

<h2 style="text-align: center;">class timetable -CSE SEM III/V </h2>

<table>

<tr>

<th> Day/Period <th>

<th> 1 <br> (9.00 – 10.00) </th>

<th> 2 <br> (10.00 – 10.50) </th>

<th> 3 <br> (11.00 – 11.50) </th>

<th> 4 <br> (11.50 – 12.40) </th>

<th> Lunch <br> Break</th>

<th> 5 <br> (1.45 – 2.30) </th>

<th> 6 <br> (2.30 – 3.15) </th>

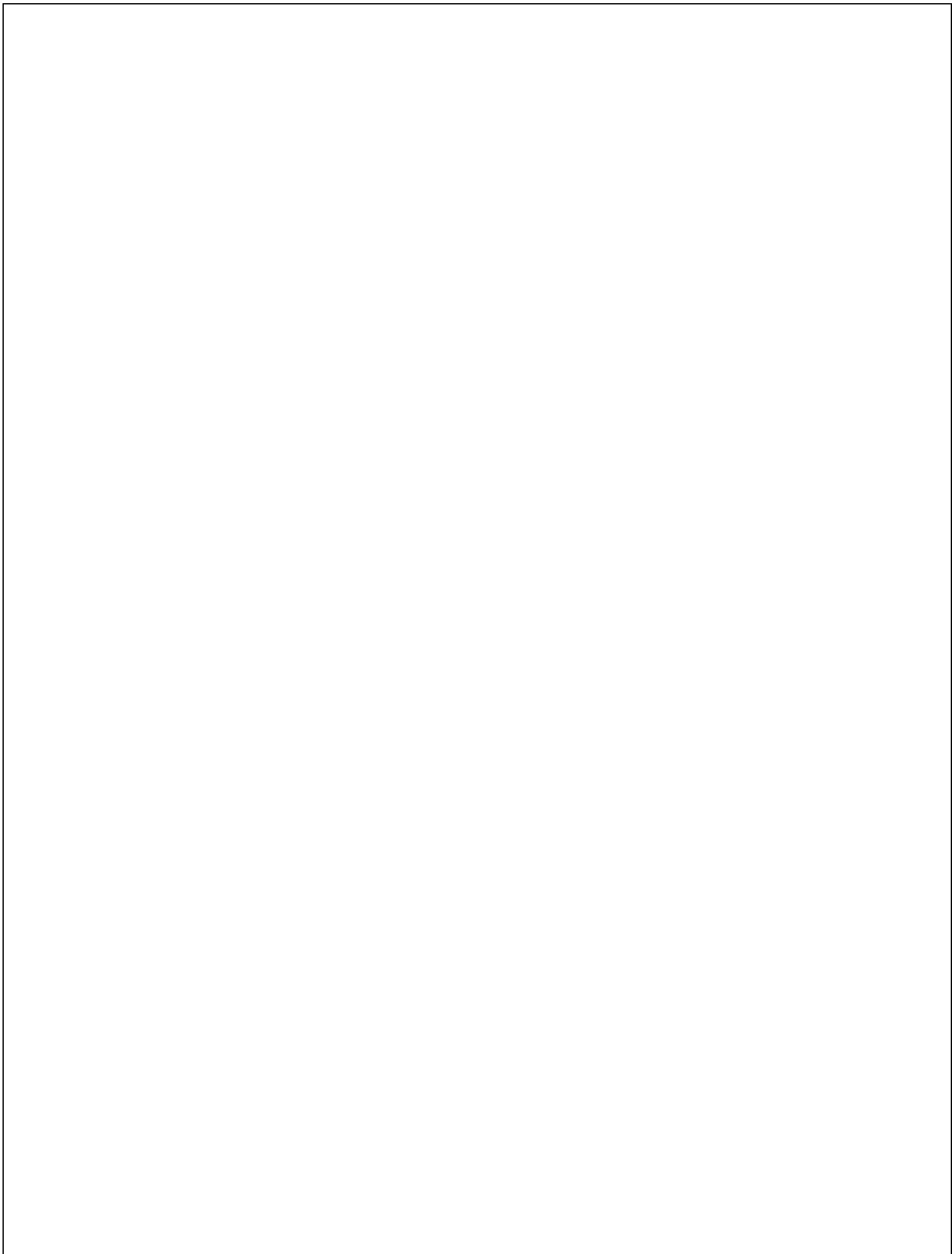
<th> 7 <br> (3.15 – 4.00) </th>

<th> 8 <br> (4.00 – 4.45) </th>

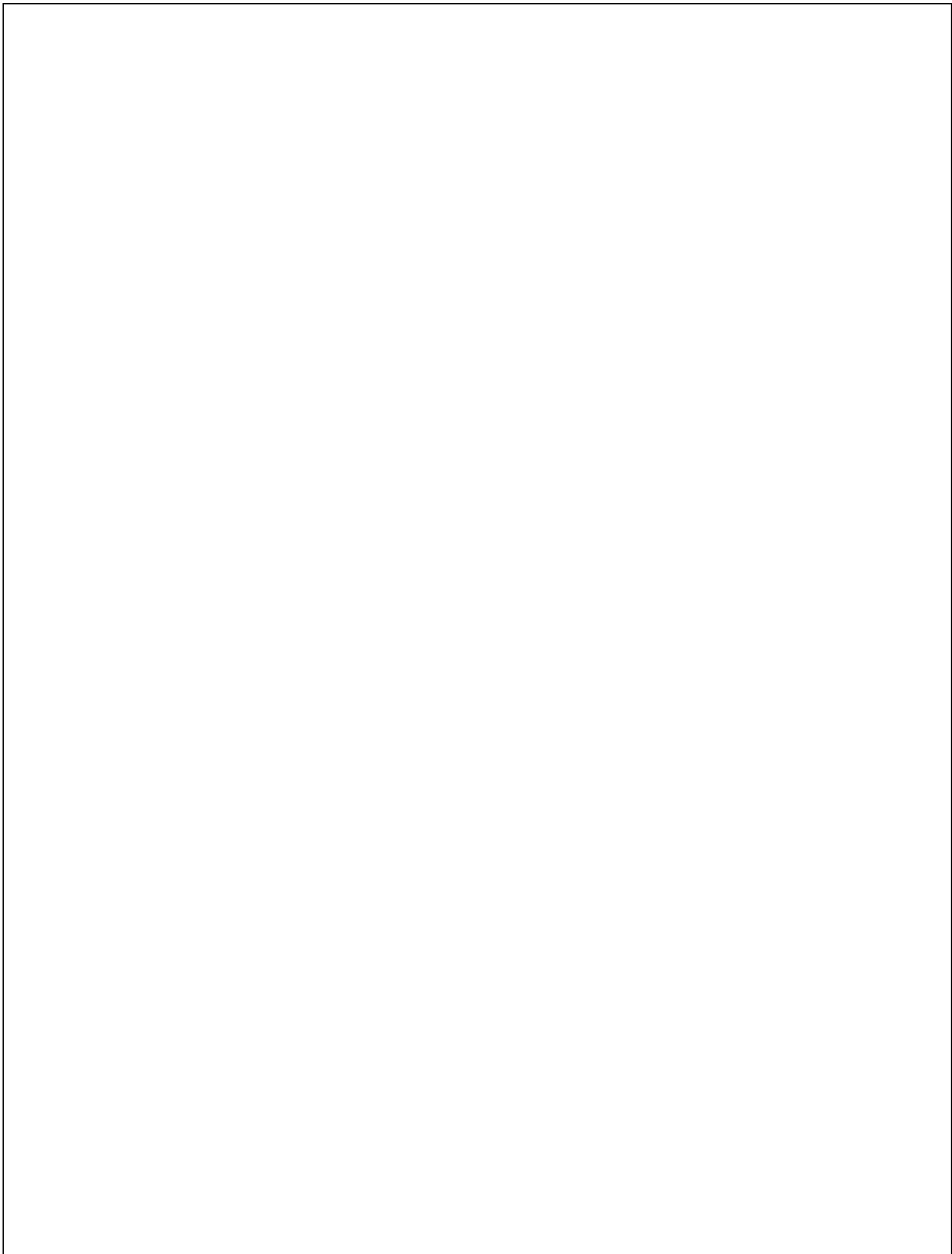
</tr>

<tr>

```



Mon	UI and UX	CN	CD	WT	Lunch Break	CCS	LIB	DC	MORAL	
</tr>										
Tue	CN	WT	DC	UI and UX	UI and UX Lab / WT Lab	UI and UX Lab / WT Lab				
</tr>										
Wed	NAAN MUDHALVAN COURSES									
</tr>										



```

<td> Thu </td>
<td>CD </td>
<td> CN </td>
<td> CCS </td>
<td> Mandatory Course </td>
<td> UI and UX </td>
<td col span="2"> Computer Networks Lab / Compiler Design </td>
<td> Placements </td>

</tr>

<td> Fri </td>
<td> DC </td>
<td> WT </td>
<td> CD </td>
<td> CCS </td>
<td> Mandatory Course </td>
<td col span="2"> Compiler Design Lab / Computer Networks Lab </td>
<td> Sports </td>

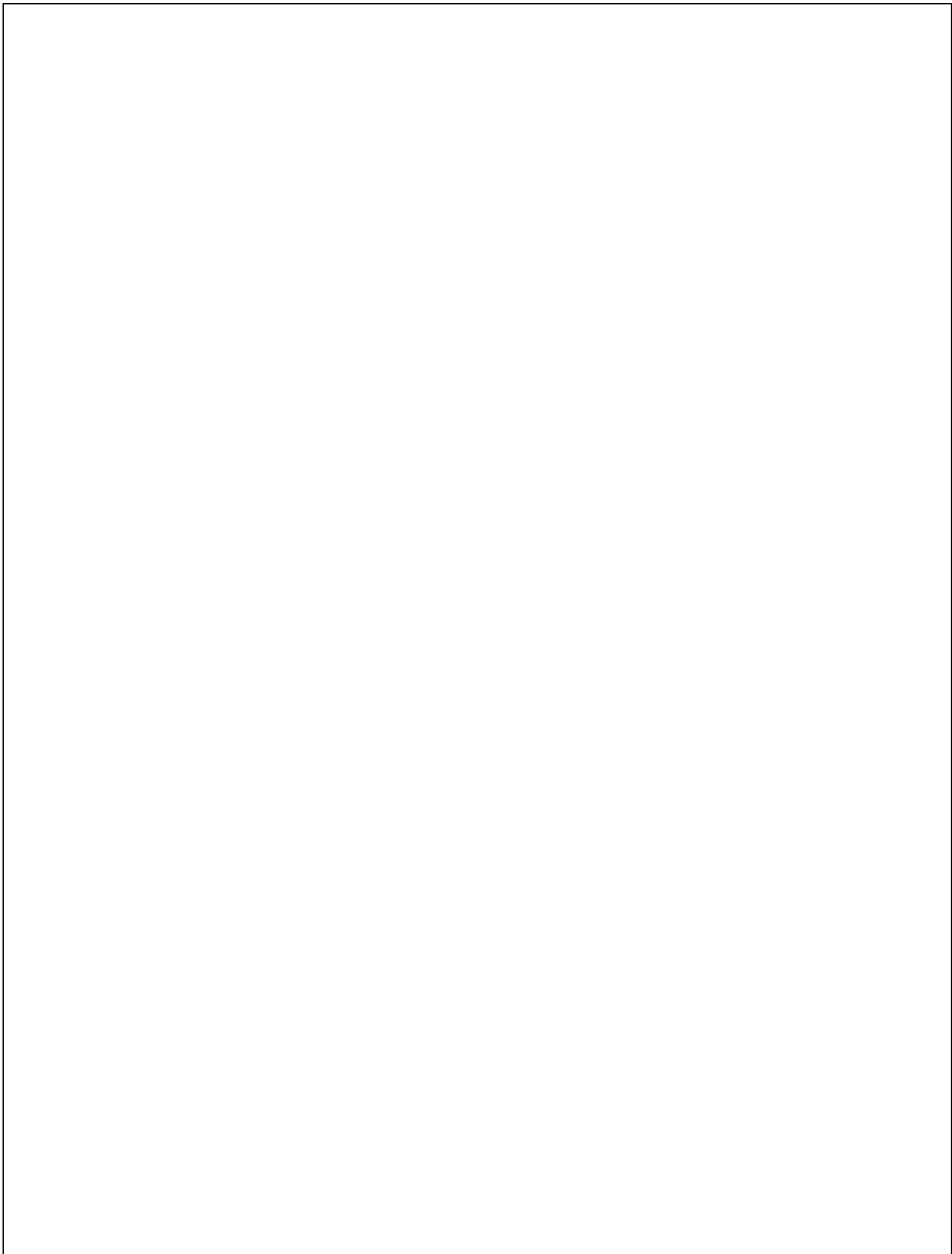
</tr>

</table>
</body>
</html>

```

RESULT:

Thus, the above program has been implemented and verified successfully.



EX NO: 03	CREATE A WEB PAGE WITH THE FOLLOWING USING HTML:-
DATE:	-TO EMBEDDED IMAGE MAP IN A WEB PAGE. -TO FIX THE HOT SPOTS. -SHOW ALL THE RELATED INFORMATION WHEN THE HOT SPOTS ARE CLICKED.

AIM:

To create web page using HTML image map that embeds an image, fixes hotspots on specific regions, and displays related information when clicked.

ALGORITHM:

- **STEP 1:** Start the HTML program with <!DOCTYPE html>.
- **STEP 2:** Insert an image (like indiamap.jpg).
- **STEP 3:** Define the <map> tag with hotspot areas using <area> and specify shape, cords, alt, and href.
- **STEP 4:** Create separate html pages (e.g., tamilnadu.html, Calcutta.html, delhi.html) with the related information.
- **STEP 5:** Add a link (Home page) in each page to return to the main map.
- **STEP 6:** Save and run the file in a browser to test hotspots.

PROGRAM:

```
//Main Page (Indiamap.html)

<!DOCTYPE html>

<html>

<head>

<title>Hotspot Creation</title>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

</head>

<body>


```

OUTPUT:

```
<map name="pic">

    <area shape="rect" coords="329,955,373,999" title="TamilNadu" alt="TamilNadu"
 href="tamilnadu.html">

    <area shape="rect" coords="652,517,695,545" title="Calcutta" alt="Calcutta"
 href="calcutta.html">

    <area shape="rect" coords="298,310,340,352" title="TamilNadu" alt="Delhi"
 href="delhi.html">

</map>

</body>

</html>
```

//Tamilnadu Page (tamil.nadu.html)

```
<html>

<body bgcolor="CYAN">

<font face="Times New Roman" size="10" color="Orange">

<center>Chennai is the capital of Tamil Nadu <br> and <br> More IT companies are camped at
Chennai</center>

<a href="Indiamap.html">Home Page</a>

</font>

</body>

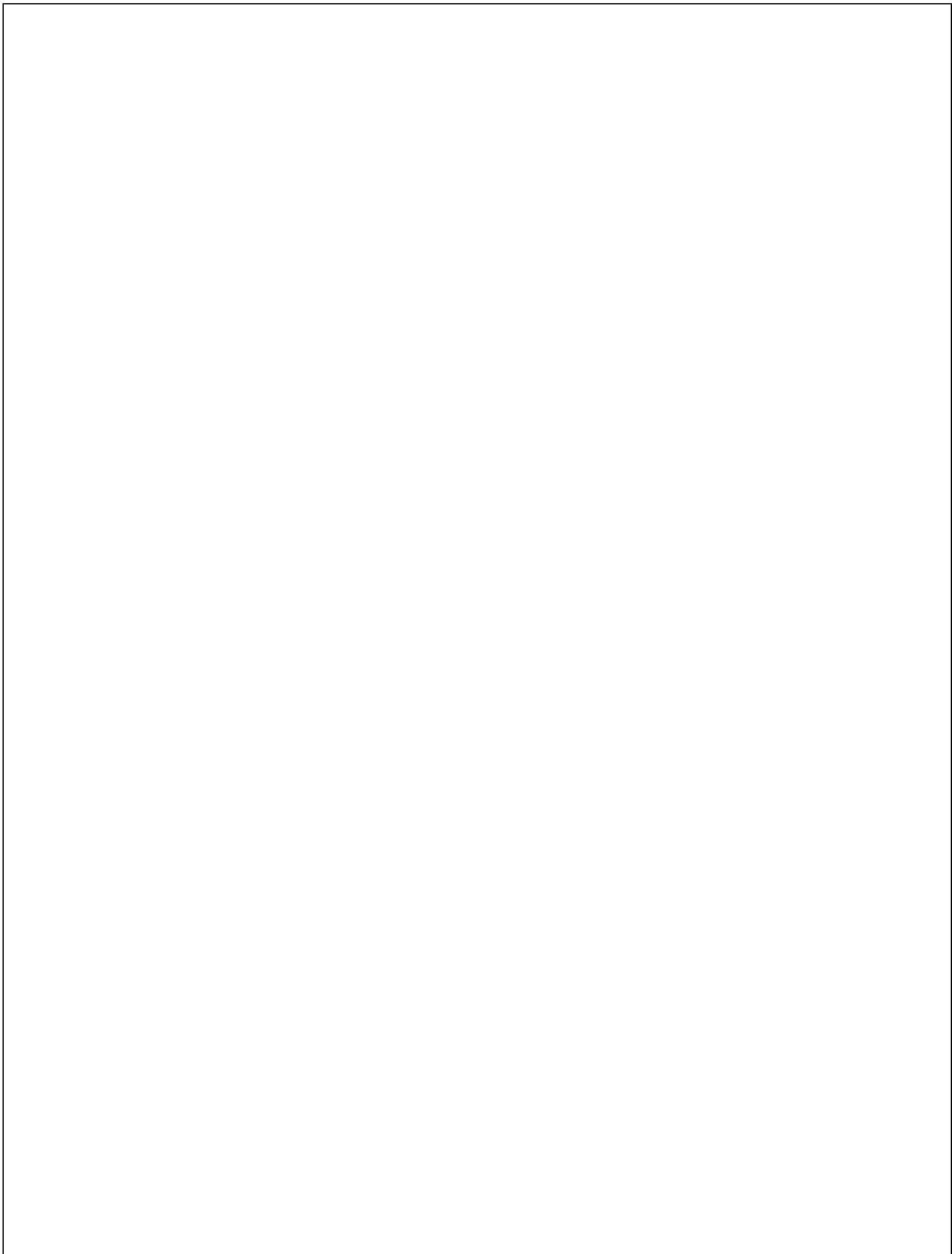
</html>
```

//Calcutta Page (Calcutta.html)

```
<html>

<body bgcolor="SKYBLUE">

<font face="Times New Roman" size="10" color="RED">
```



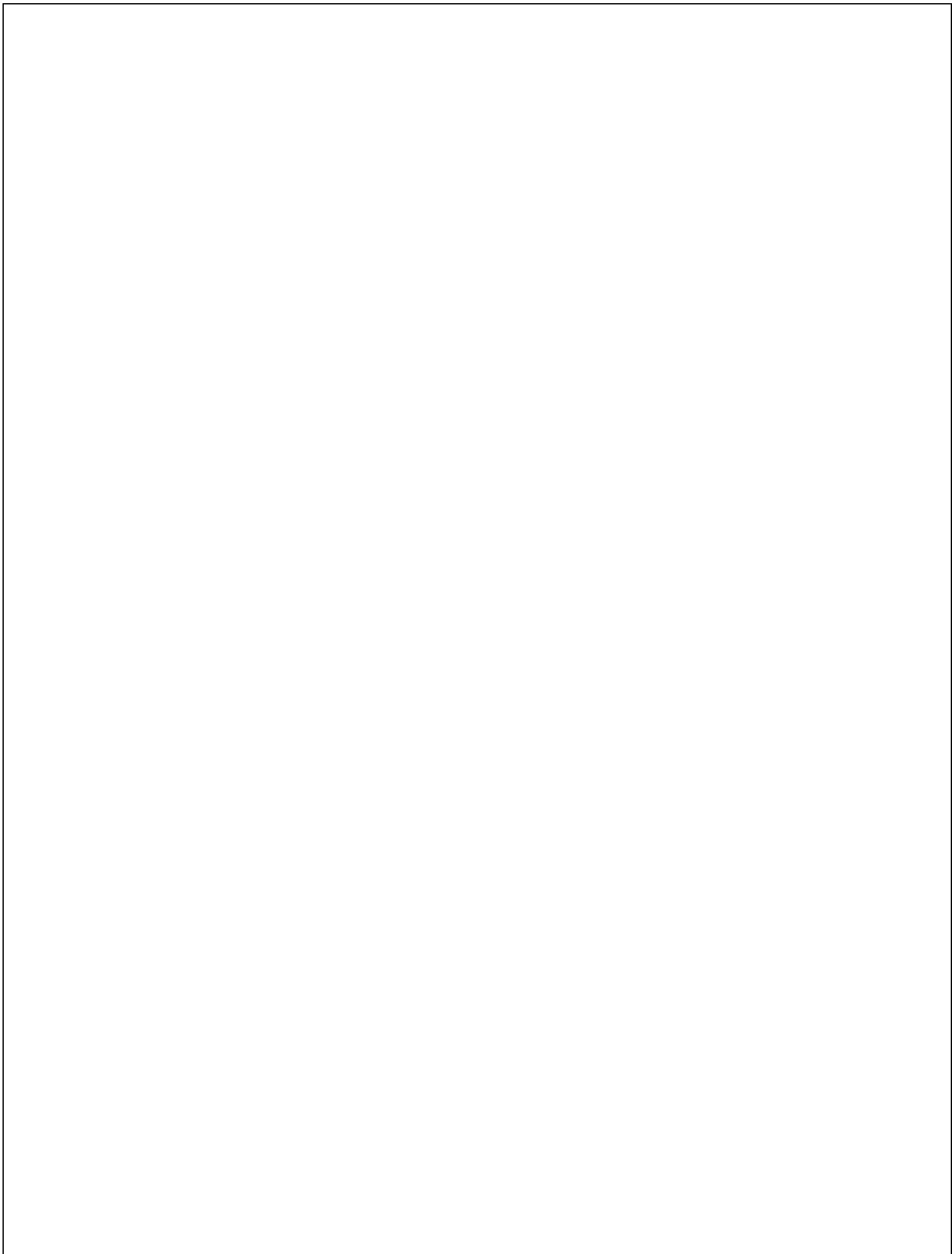
```
<center><b><i>Calcutta is the wealthy city in WEST BENGAL <br> and <br> It has Famous  
"Sunderbans Forests"</i></b></center>  
  
<a href="Indiamap.html">Home Page</a>  
  
</font>  
  
</body>  
  
</html>
```

//Delhi Page (delhi.html)

```
<html>  
  
<body bgcolor="SKYBLUE">  
  
<font face="Arial" size="10" color="RED">  
  
<center><b><i><tt>Delhi is the capital of our INDIA <br> and <br> More IT companies are  
camped at Delhi </tt></i></b></center>  
  
<a href="Indiamap.html">Home Page</a>  
  
</font>  
  
</body>  
  
</html>
```

RESULT:

Thus the above program creation of web page for India map using html has been implemented and verified successfully.



Ex no:04

Date:

CREATE A WEBPAGE WITH ALL TYPES OF CASCADING STYLE SHEETS

AIM:

To create a webpage that demonstrate all types of CSS, Inline CSS, Internal CSS and External CSS.

ALGORITHM:

- **STEP 1:** Start by cracking an HTML file and name it index.html.
- **STEP 2:** Write the basic HTML structure <!DOCTYPE html>, <html>, <head>, <body>.
- **STEP 3:** Add internal CSS inside the <style's> tag in the <head> section.
- **STEP 4:** Create an external CSS file and link it using <link>, rel= “stylesheet” href= “style.css”.
- **STEP 5:** Use inline CSS by adding the style attribute directly inside HTML tags.
- **STEP 6:** Add sample text, headings, paragraphs, styled with each type of CSS.
- **STEP 7:** Save the files and open index.html in a browser to view the result.
- **STEP 8:** Stop the program.

PROGRAM:

```
//index.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>All Types of CSS</title>

    <style>

        body {

            background-color: #f0f0f0;
```

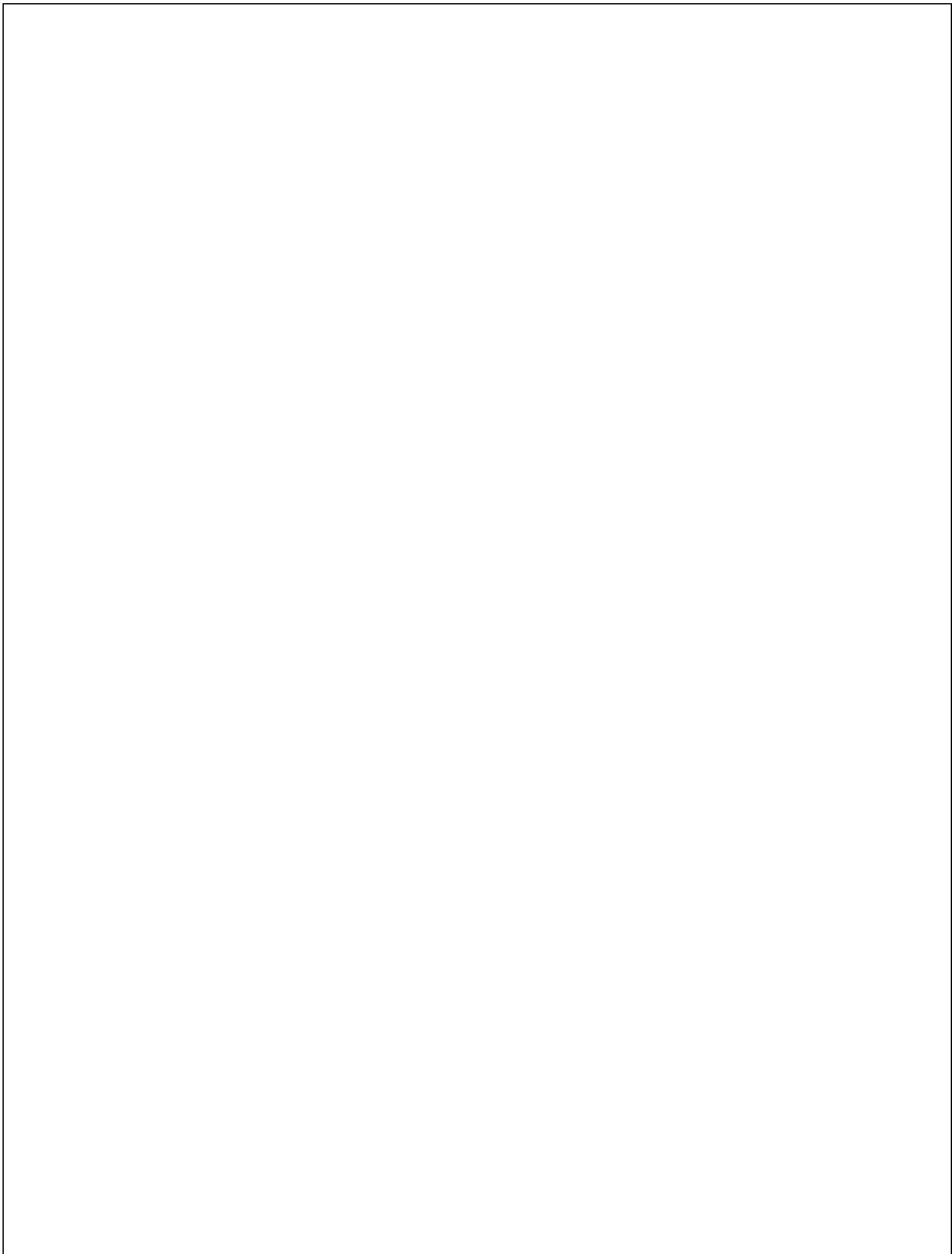
OUTPUT:

```
font-family: Arial, sans-serif;  
}  
.internal {  
    color: blue;  
    font-size: 18px;  
}  
</style>  
<link rel="stylesheet" href="style.css">  
</head>  
<body>  
    <h1 style="color: green; text-align: center;">  
        Inline CSS Example  
    </h1>  
    <p class="internal">  
        This paragraph is styled using Internal CSS.  
    </p>  
    <p class="external"> This paragraph will be styled using External CSS.</p>  
</body>  
</html>  
//External CSS File (style.css):  
.external-style {  
    color: darkred;  
    font-size: 20px;  
    font-weight: bold;
```

}

RESULT:

Thus the creation of webpage for all types of cascading style sheets are implemented and verified successfully.



Ex no: 05

Date:

CLIENT SIDE SCRIPTS FOR VALIDATING WEB FORM CONTROLS USING DHTML

AIM:

To design a student registration form and validate form fields (Name, Email, Password, Age, Gender, Course and Terms) using DHTML (HTML + JavaScript on the client side).

ALGORITHM:

- **STEP 1:** Start the program.
- **STEP 2:** Create a registration form using HTML from elements (text, password, radio, combo box, checkbox).
- **STEP 3:** Write a JavaScript function validation form for validation.
- **STEP 4:** Check conditions;
 1. Name should not be empty.
 2. Email must be in correct format,
 3. Password must have atleast (characters).
 4. Age must be a positive number.
 5. Gender must selected.
 6. Course must be selected from the list.
 7. Terms and conditions must be accepted.
- **STEP 5:** If any validation fails, then it shows an error message red.
- **STEP 6:** If all validation pass, then it shows “Form successfully submitted!” in green.

PROGRAM:

```
<!DOCTYPE html>

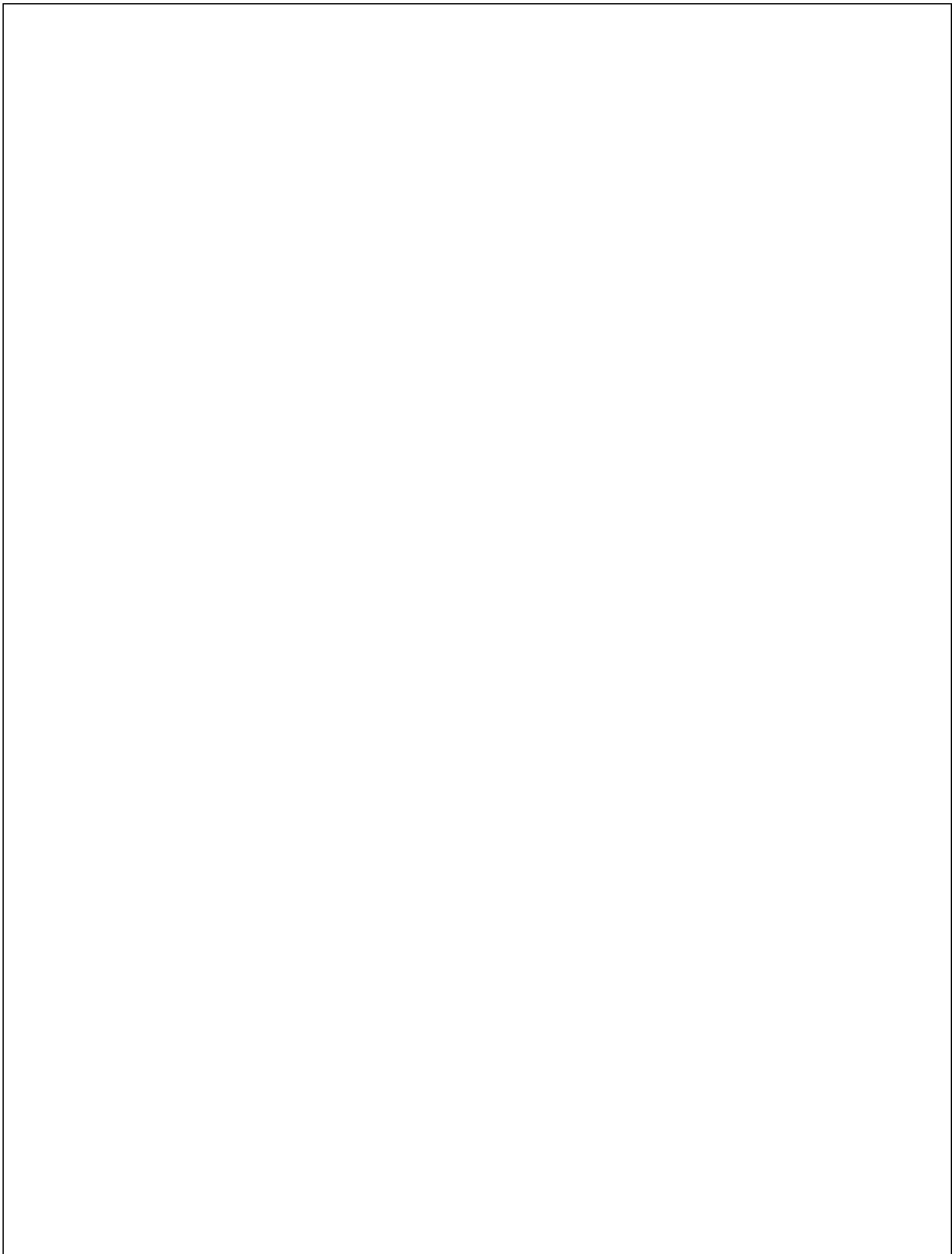
<html>
<head>
<title>Form Validation using DHTML</title>
<script>

function validateForm() {

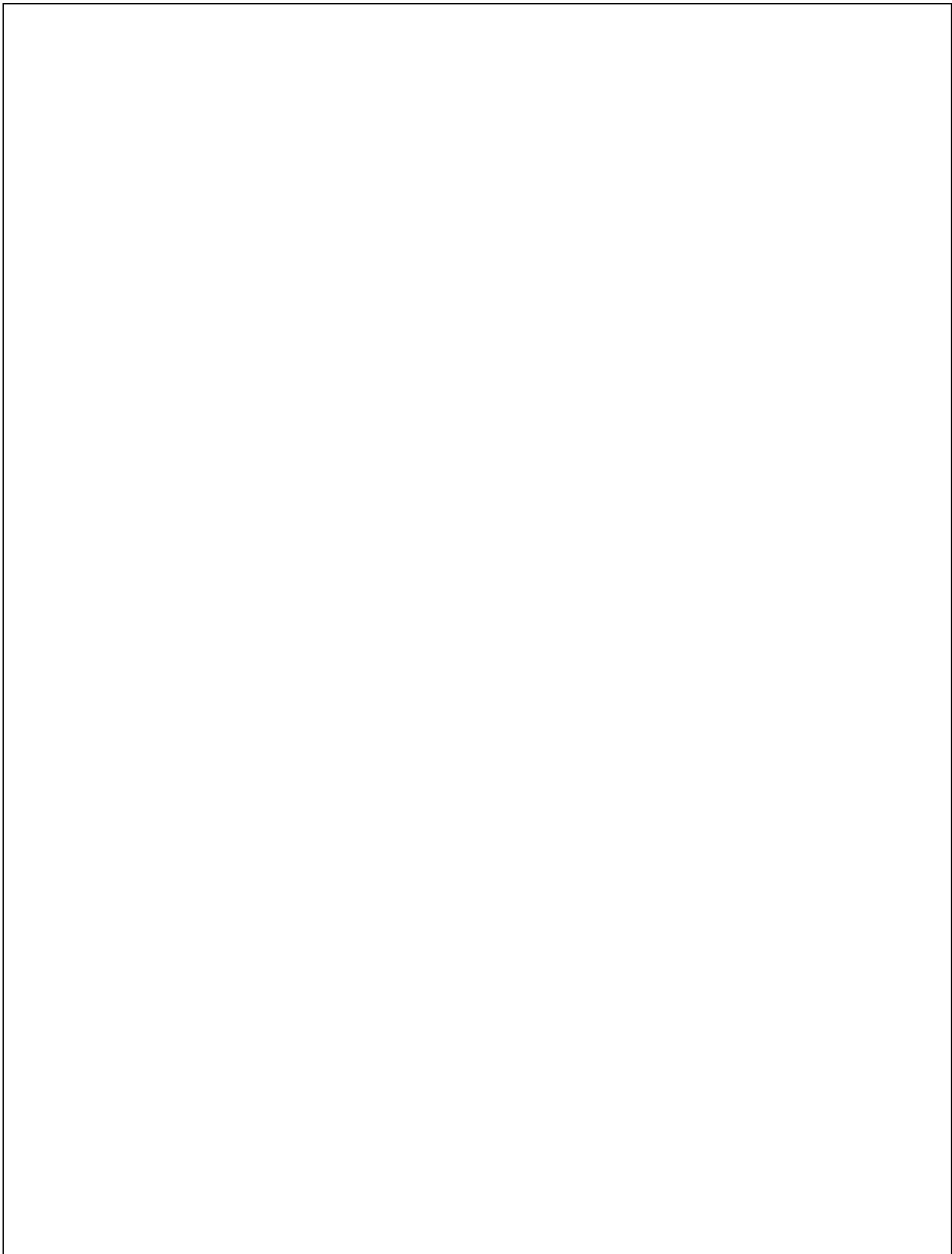
    var name = document.getElementById("name").value;
```

OUTPUT:

```
var email = document.getElementById("email").value;
var password = document.getElementById("password").value;
var age = document.getElementById("age").value;
var gender = document.querySelector('input[name="gender"]:checked');
var course = document.getElementById("course").value;
var terms = document.getElementById("terms").checked;
var msg = "";
var successMsg = document.getElementById("success");
var errorMsg = document.getElementById("error");
successMsg.innerHTML = ""; // Clear old success
errorMsg.innerHTML = ""; // Clear old errors
if (name.trim() === "") {
    msg += "Name cannot be empty.<br>";
}
var emailPattern = /^[^ ]+@[^ ]+\.[a-z]{2,3}$/;
if (!email.match(emailPattern)) {
    msg += "Enter a valid email address.<br>";
}
if (password.length < 6) {
    msg += "Password must be at least 6 characters.<br>";
}
if (isNaN(age) || age <= 0) {
    msg += "Enter a valid age.<br>";
}
if (!gender) {
```



```
msg += "Please select your gender.<br>";  
}  
  
if (course === "select") {  
  
    msg += "Please select a course.<br>";  
}  
  
if (!terms) {  
  
    msg += "You must accept the terms and conditions.<br>";  
}  
  
if (msg !== "") {  
  
    errorMsg.innerHTML = msg; // Show errors in red  
  
    return false; // Prevent submission  
} else {  
  
    successMsg.innerHTML = " ✅ Form Successfully Submitted!";  
  
    return false; // Keep page from refreshing  
} }  
  
</script>  
  
</head>  
  
<body>  
  
<h2>Student Registration Form</h2>  
  
<form onsubmit="return validateForm()">  
  
Name: <input type="text" id="name"><br><br>  
  
Email: <input type="text" id="email"><br><br>  
  
Password: <input type="password" id="password"><br><br>  
  
Age: <input type="text" id="age"><br><br>
```



```

Gender:
<input type="radio" name="gender" value="Male"> Male
<input type="radio" name="gender" value="Female"> Female
<br><br>

Course:
<select id="course">
    <option value="select">--Select Course--</option>
    <option value="B.Tech">B.Tech</option>
    <option value="M.Tech">M.Tech</option>
    <option value="MCA">MCA</option>
    <option value="MBA">MBA</option>
</select><br><br>

<input type="checkbox" id="terms"> I accept the Terms and Condition
<br><br>

<input type="submit" value="Submit">
<input type="reset" value="Reset">

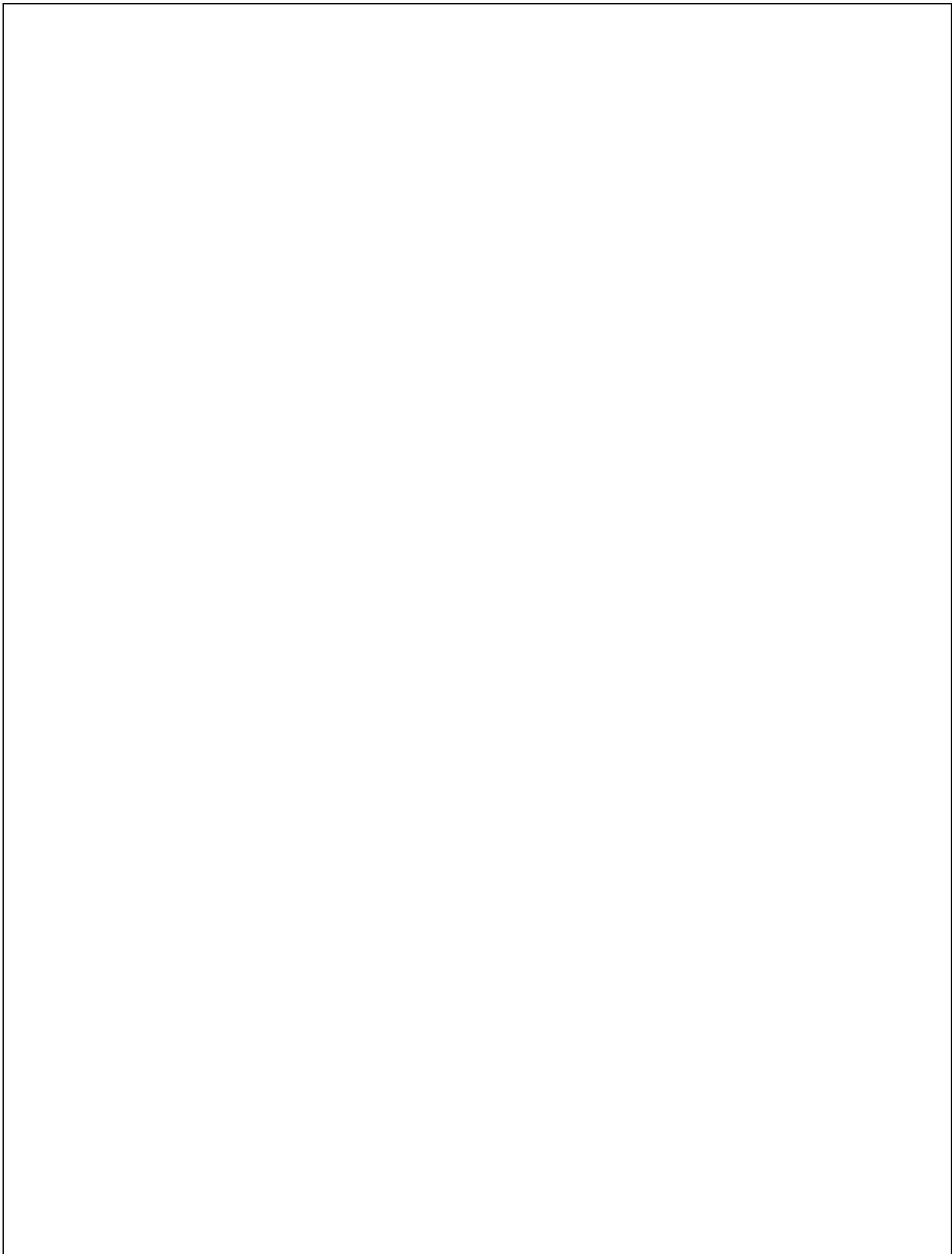
</form><div id="error" style="color:red; font-weight:bold;"></div>
<div id="success" style="color:green; font-weight:bold;"></div>

</body>
</html>

```

RESULT:

Thus the creation of client side scripts for validating web form controls using DHTML was implemented and verified successfully.



Ex no: 06

INSTALLATION OF APACHE TOMCAT USING WEB SERVER

AIM:

To install and configure Apache tomcat web server on window operating system for running java-based web applications.

ALGORITHM:

- **STEP 1:** Start the system and ensure java JDK is installed.
- **STEP 2:** Set the download JAVA_HOME environment variable for java.
- **STEP 3:** Download the latest version of Apache tomcat from the official website.
- **STEP 4:** Extract the downloaded ZIP file into a suitable director (eg: C:\apache-tomcat-9.0.109).
- **STEP 5:** Set CATALINA_HOME environment variable to point to the Tomcat directory.
- **STEP 6:** Add %CATALINA_HOME% / bin to the system PATH variable.
- **STEP 7:** Run the startup.bat file from the Tomcat bin folder.
- **STEP 8:** Open a web browser and access <http://localhost:8080> to verify installation.
- **STEP 9:** Run shutdown.bat to stop the Tomcat server.

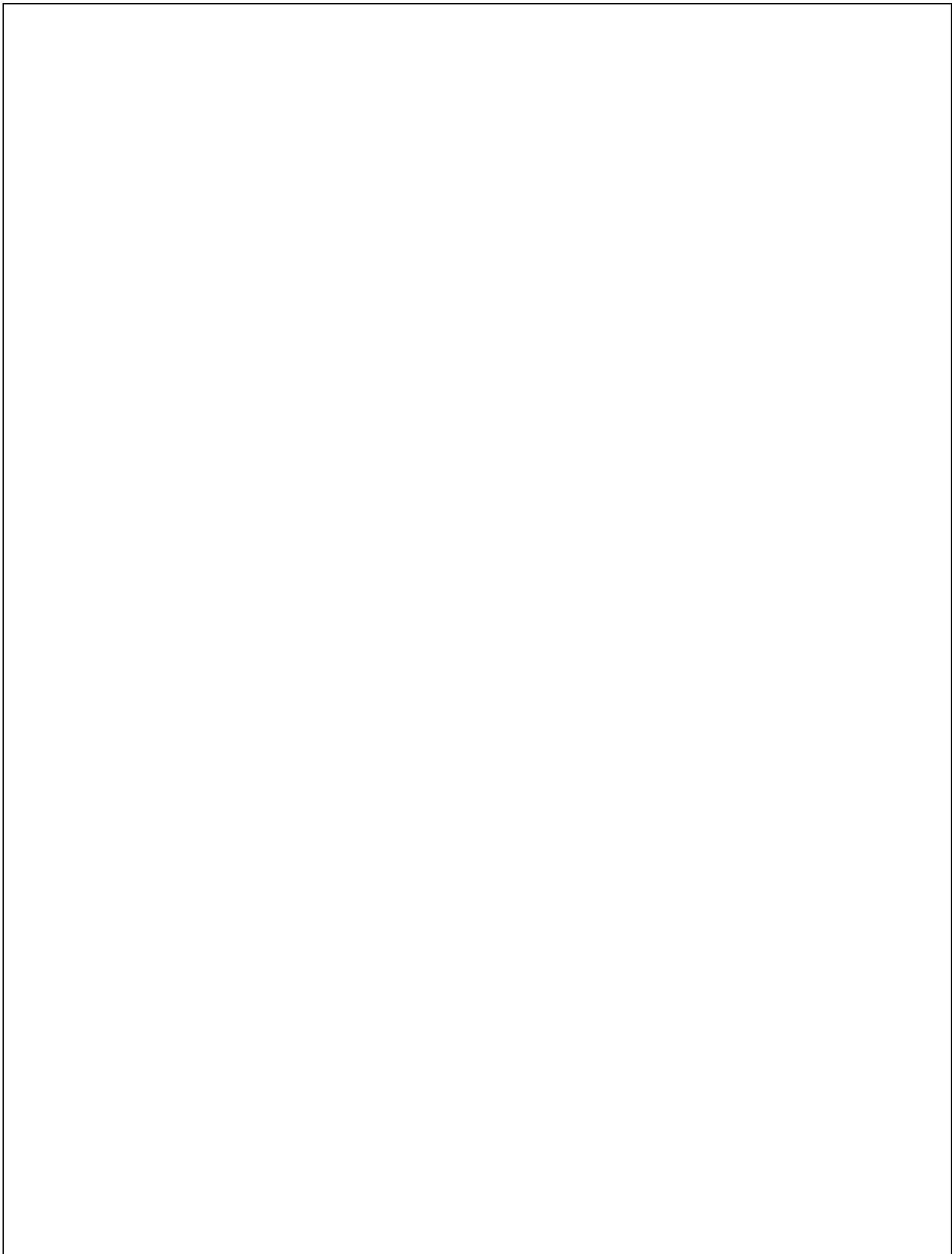
PROCEDURE:

- Download and install Java JDK from oracle / Adopt open JDK.
- Set JAVA-HOME in environment variables to the JDK installation path.
- Verify installation using command prompt as; java –version.
- Visit <https://tomcat.apache.org/> and download the windows ZIP file of Apache Tomcat.
- Extract the ZIP file to C:\apache-tomcat-9.0.109.
- Open system properties → Environment variables.
 - ✓ Add new variables: CATALINA_HOME = C:\apache-tomcat-9.0.109.
 - ✓ Edit path → add % CATALINA_HOME % \ bin.
- Open command prompt, go to C:\apache-tomcat-9.0.109\bin and run: Startup.bat.
- Open a web browser and enter <http://localhost:8080>.
 - ✓ The Apache Tomcat welcome page should be displayed.
- To stop the server, run: Shutdown.bat.

OUTPUT:

RESULT:

Thus, the above installation of Apache Tomcat web server on windows installed successfully.



Ex no: 07	WRITE PROGRAMS IN JAVA USING SERVLETS
Date:	1) TO INVOKE SERVLETS FROM HTML FORMS 2) SESSION TRACKING

AIM:

To write a Java servlet program to:

1. Invoke a servlet from an HTML form.
2. Track user session using Http Session.
3. Display the session details and number of visits.

ALGORITHM:

- **STEP 1:** Start the program.
- **STEP 2:** Create an HTML form to accept user details like Name, ID, Email, Phone, City.
- **STEP 3:** The form send data to a servlet using the POST method.
- **STEP 4:** The servlet receives the form data using request. getSession().
- **STEP 5:** Create or get an existing session using request.getSession().
- **STEP 6:** Store user data in the session attributes.
- **STEP 7:** Count how many times the user visited the page using session counter.
- **STEP 8:** Display the user details and the session visit count in the browser.
- **STEP 9:** Stop the program.

PROGRAM:

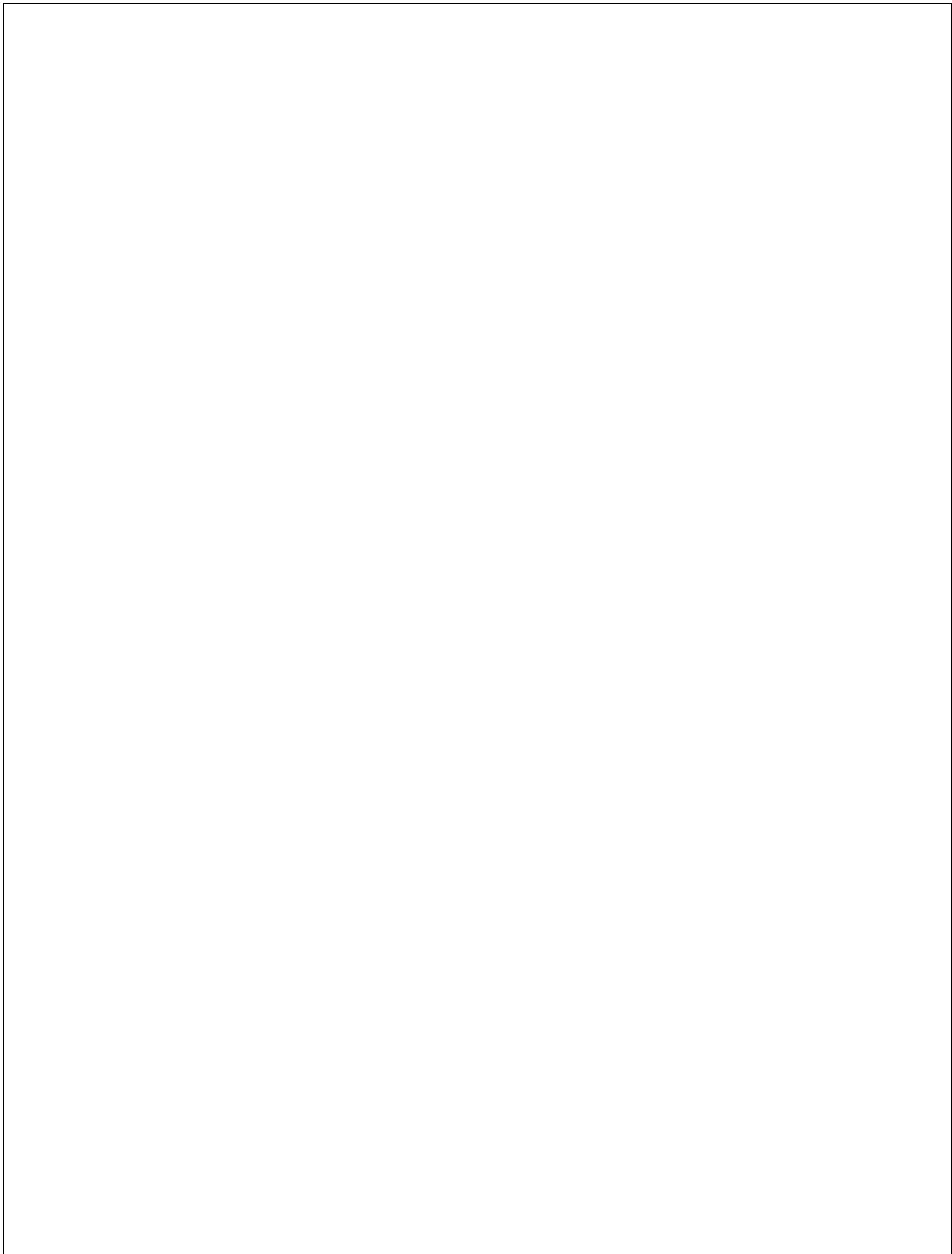
```
//index.html

<!DOCTYPE html>

<html>
<head>
<title>Web Form</title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<style>
```

OUTPUT:

```
. heading{  
    Padding-left:400px;  
}  
. submit{  
    Padding-left:500px;  
}  
</style>  
</head>  
<body>  
<form method ="post" action ="New Servlet">  
<label class ="heading"> Name: <input type ="text" name ="id"> </label> <br >  
<label class ="heading"> Id: <input type ="text" name ="id"> </label> <br>  
<label class ="heading"> Email: <input type ="email" name ="email"> </label> <br>  
<label class ="heading"> Phone Number: <input type ="text" name ="Phno"> </label> <br>  
<label class ="heading"> City: <input type ="text" name ="city"> </label> <br>  
<label class ="Submit"> <input type= "Submit" value="Submit">  
</label>  
</form>  
</body>  
</html>
```



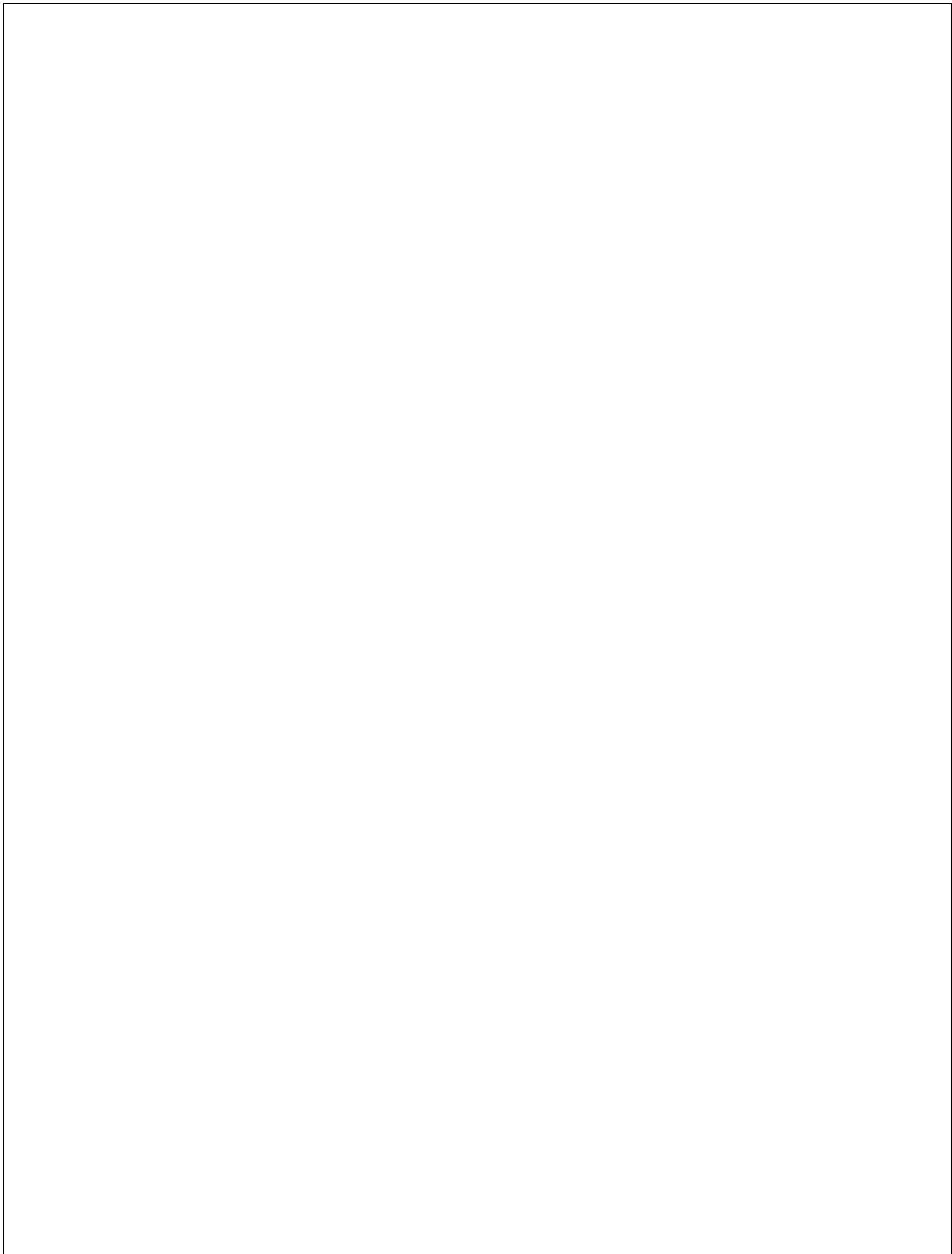
```
//Servlet.java

import java.io.IOException;
import java.io.PrintWriter;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;

Public class NewServlet extends HttpServlet{

@Override
public void service (ServletRequest request, ServletResponse response){
    response.setContentType("text/html");
    PrintWriter out =null;
    try{
        out =response.getWriter();
    }catch (IOException ex){
        Logger.getLogger (NewServlet.class.getname()).log(Level.SEVERE, null, ex);
    }
    String name = request.getParameter("name");
    String id = request.getParameter("id");
    String email= request.getParameter("email");
    String phno = request.getParameter("city");

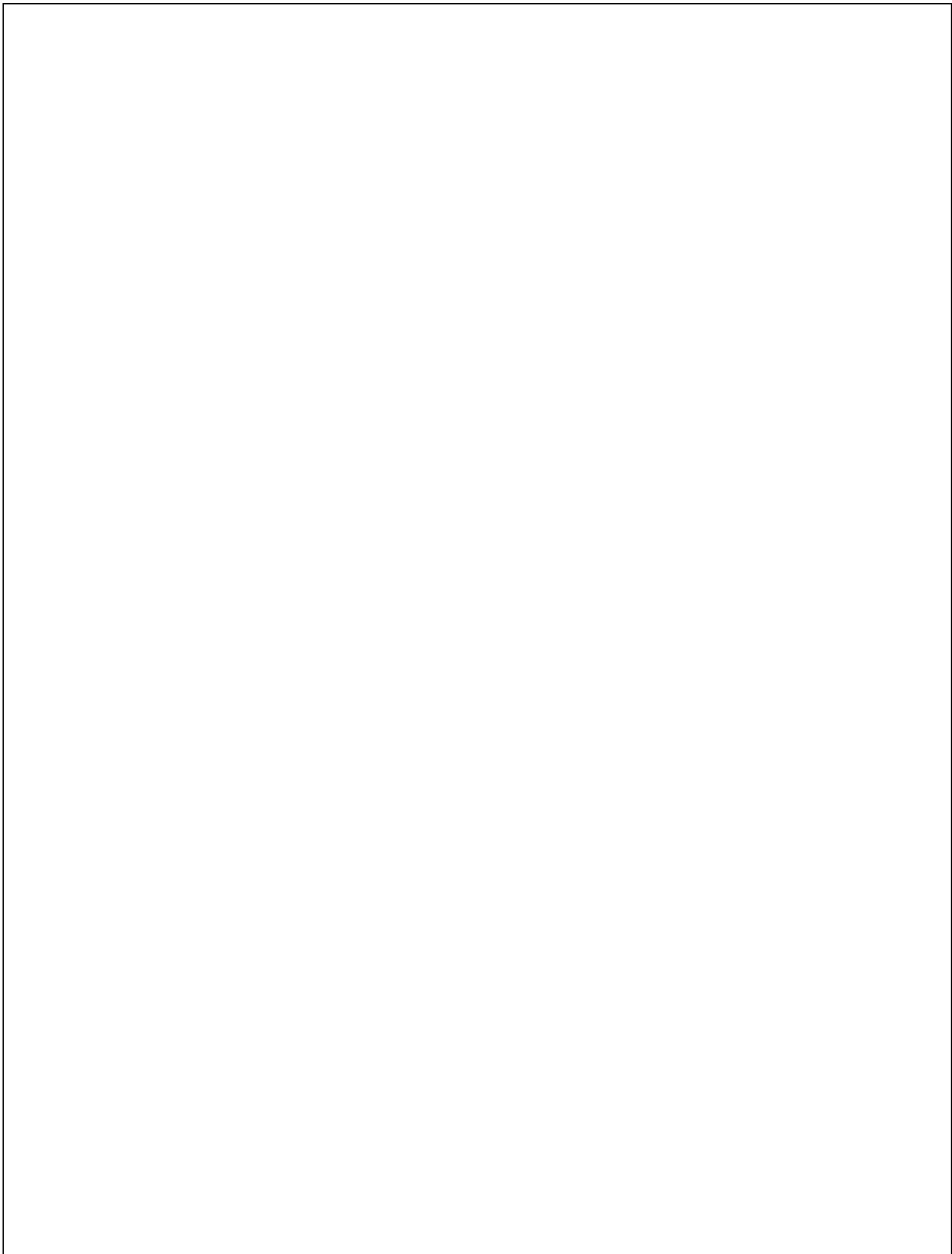
    Out.println("Name:"+name);
    Out.println("Id:"+id);
    Out.println("Email:"+email);
    Out.println("Phno:"+phno);
```



```
Out.println("City:"+city);  
}  
}
```

RESULT:

Thus, the above program has been implemented and verified successfully.



Ex no: 08

**WRITE PROGRAMS IN JAVA TO CREATE THREE-TIER
APPLICATION USING SERVLETS FOR CONDUCTING
ONLINE EXAMINATION FOR DISPLAYING
STUDENT MARK LIST**

Date:

AIM:

Write programs in Java to create three-tier applications using servlets for conducting On-line examination for displaying student mark list. Assume that student information is available in a database which has been stored in a database server.

ALGORITHM:

STEP 1: Start the program.

STEP 2: Create a database named OnlineExamDB.

STEP 3: Create the following tables in the database:

- Student (Student_id, name, username, password)
- Questions (q_id, question, option1, option2, option3, option4, correct_option)
- Marks (Student_id, marks)

STEP 4: Design the front-end pages (HTML /JSP):

- Login Page
- Exam Page
- Result Page

STEP 5: Create a LoginServlet to:

- Get the username and password from the login page.
- Check the credentials from the database using DAO.
- If valid → move to Exam Page.
- Else → show "Invalid Login" message.

OUTPUT:

STEP 6: Create an ExamServlet to:

- Fetch questions from the database.
- Display them to the student.
- Get the student's answers.
- Compare answers with the correct ones and calculate marks.

STEP 7: Create a ResultServlet to:

- Store the Student's marks in the database.
- Display the marks on the result page.

STEP 8: Create a DAO class to handle all database operations using JDBC.

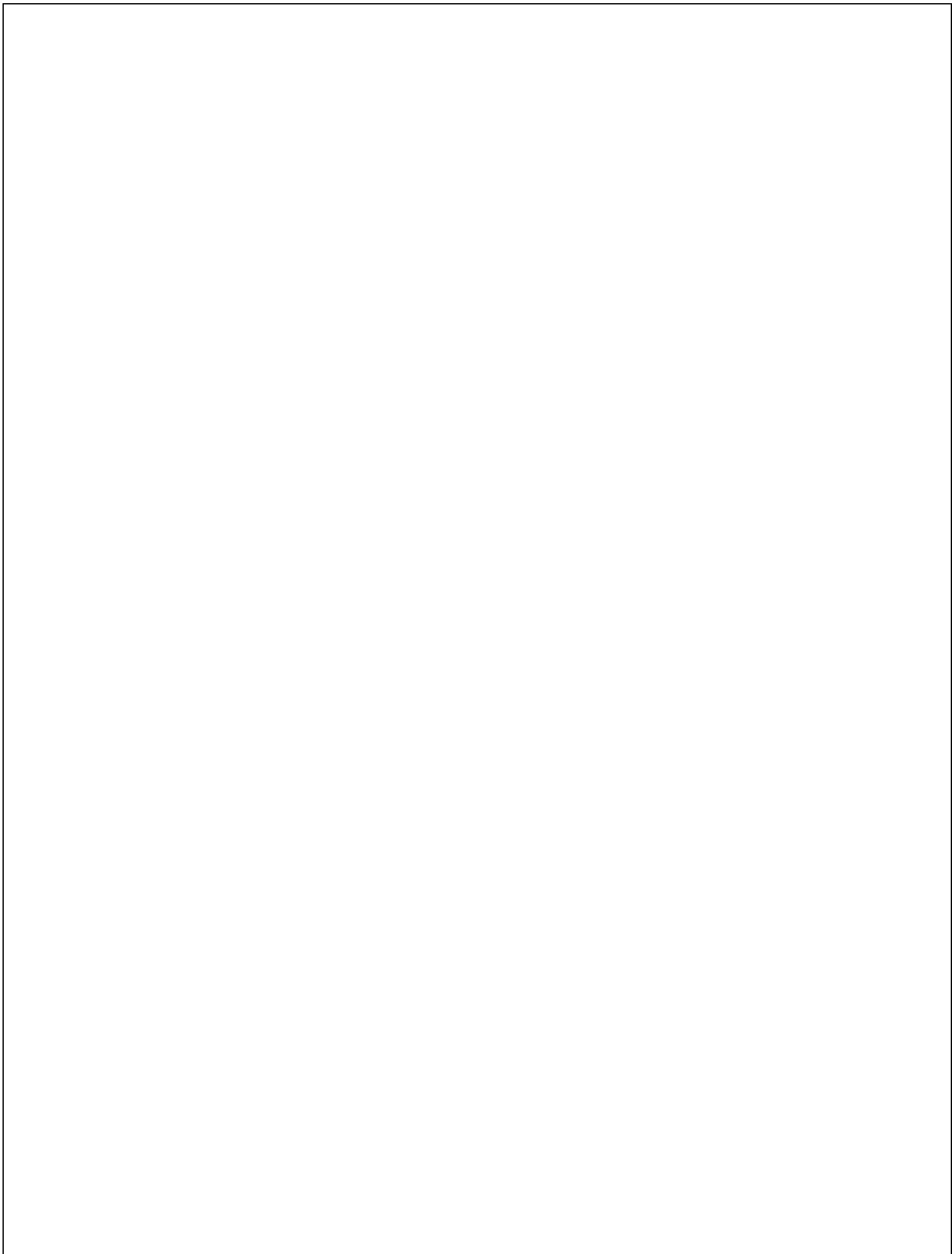
STEP 9: Close the database connection.

STEP 10: End the program.

PROGRAM:

Index.html

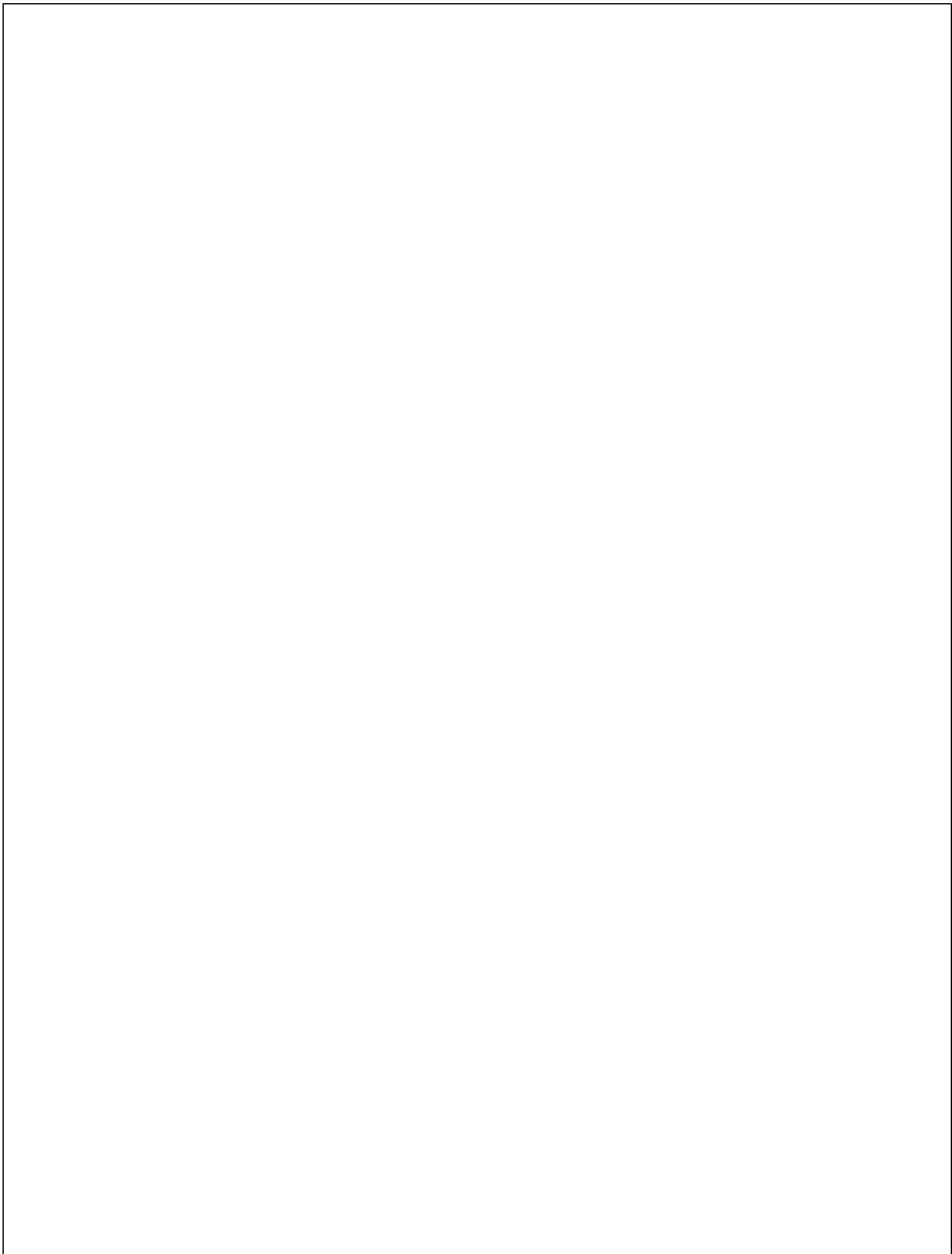
```
<html>
<head>
<title>exam portal</title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
<div style="text-align: center">
<h3>ONLINE EXAM PORTAL -Login<h3>
<hr>
<form method="get" action="acceptuser.jsp">
Username<input type="text" name="uname" value=""><br><br>
Password<input type="password" name="pass" value=""><br><br>
```



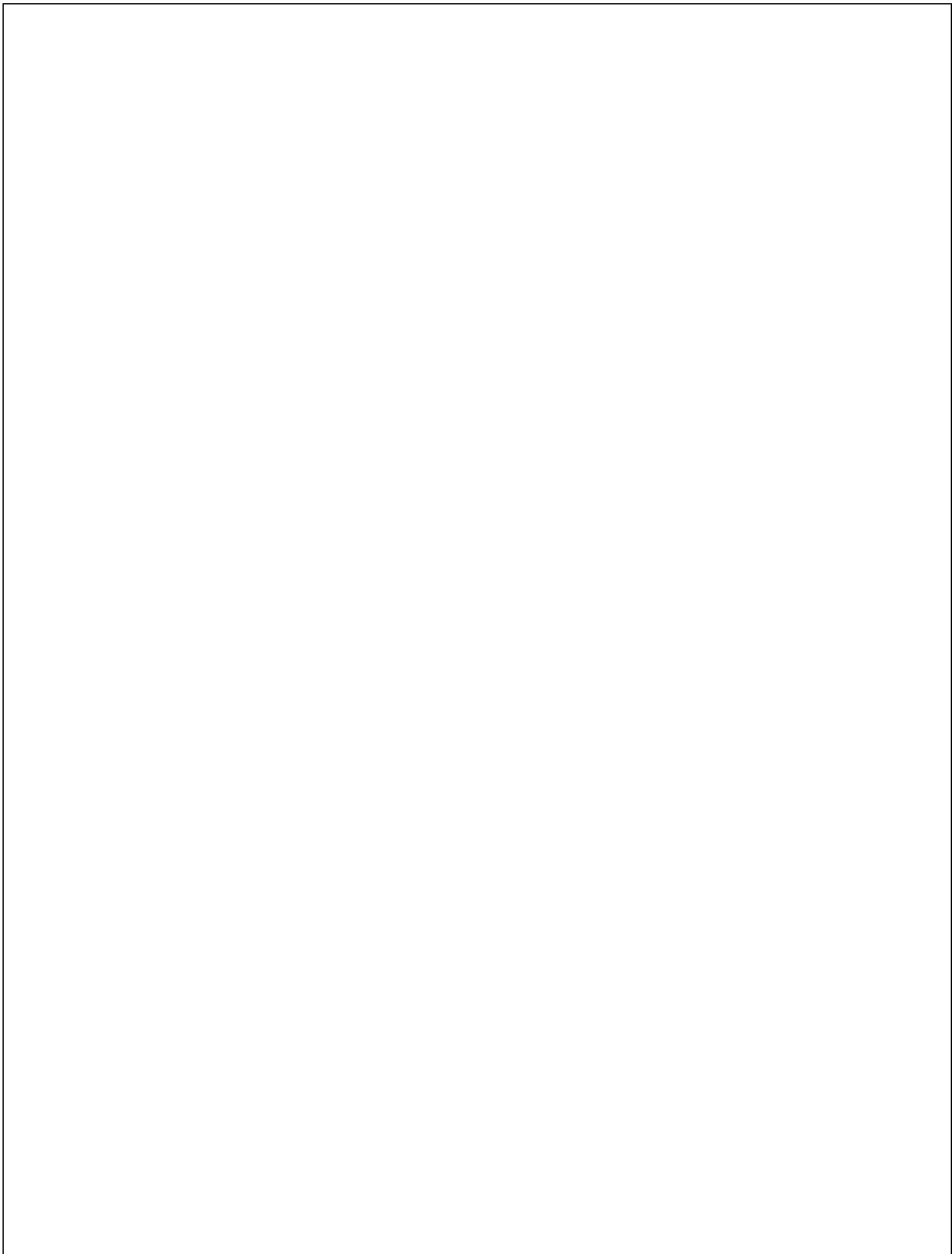
```
<button type="submit">LOGIN</button>
</form>
</div>
</body>
</html>
```

Acceptuser.jsp

```
<%@ page import="java.sql.*"%>
<%@ page import="java.util.*"%>
<%!
Connection con;
PreparedStatement ps1, ps2;
public void jspInit()
{
try{
Class.forName("org.apache.derby.jdbc.ClientDriver").newInstance();
con=DriverManager.getConnection("jdbc:derby://localhost:1527/iplab","root","root");
ps1=con.prepareStatement("select count(*)from USERS where username=?and password=?");
ps2 =con.prepareStatement("select*from USERS");
}
catch(Exception ex)
{
ex.printStackTrace();
}
}
%>
```



```
<%  
String param = request.getParameter("s1");  
If(param == "link")  
{  
ResultSet rs = ps2.executeQuery();  
out.println("<table>");  
While(rs.next())  
{  
out.println("<tr>");  
out.println("<td>" + rs.getString(1) + "</td>");  
out.println("<td>" + rs.getString(2) + "</td>");  
out.println("</tr>");  
}  
out.println("</table>");  
rs.close();  
}  
else{  
String user = request.getParameter("uname");  
String pass = request.getParameter("pass");  
ps1.setString(1, user);  
ps1.setString(2, pass);  
resultSet rs = ps1.executeQuery();  
int cnt = 0;  
if(rs.next())  
cnt = rs.getInt(1);
```



```
If(cnt == 0)

    out.println("<b><i><font color=red>Invalid credential</font></i></b>");

else{

    out.println("<form><center><fieldset style = width:55% height:60%;>");

    out.println("<b><font color=red>Valid Credential..</font></b><br>");

    out.println("<b><a href = examclient.html><font size = 6 color = blue>Click Here to take
Online Exam</font></a></b>");

    out.println("</fieldset></form>");

}

}

%>

<%!

Public void jspDestroy()

{

try{

ps1.close();

ps2.close();

con.close();

}

catch(Exception ex)

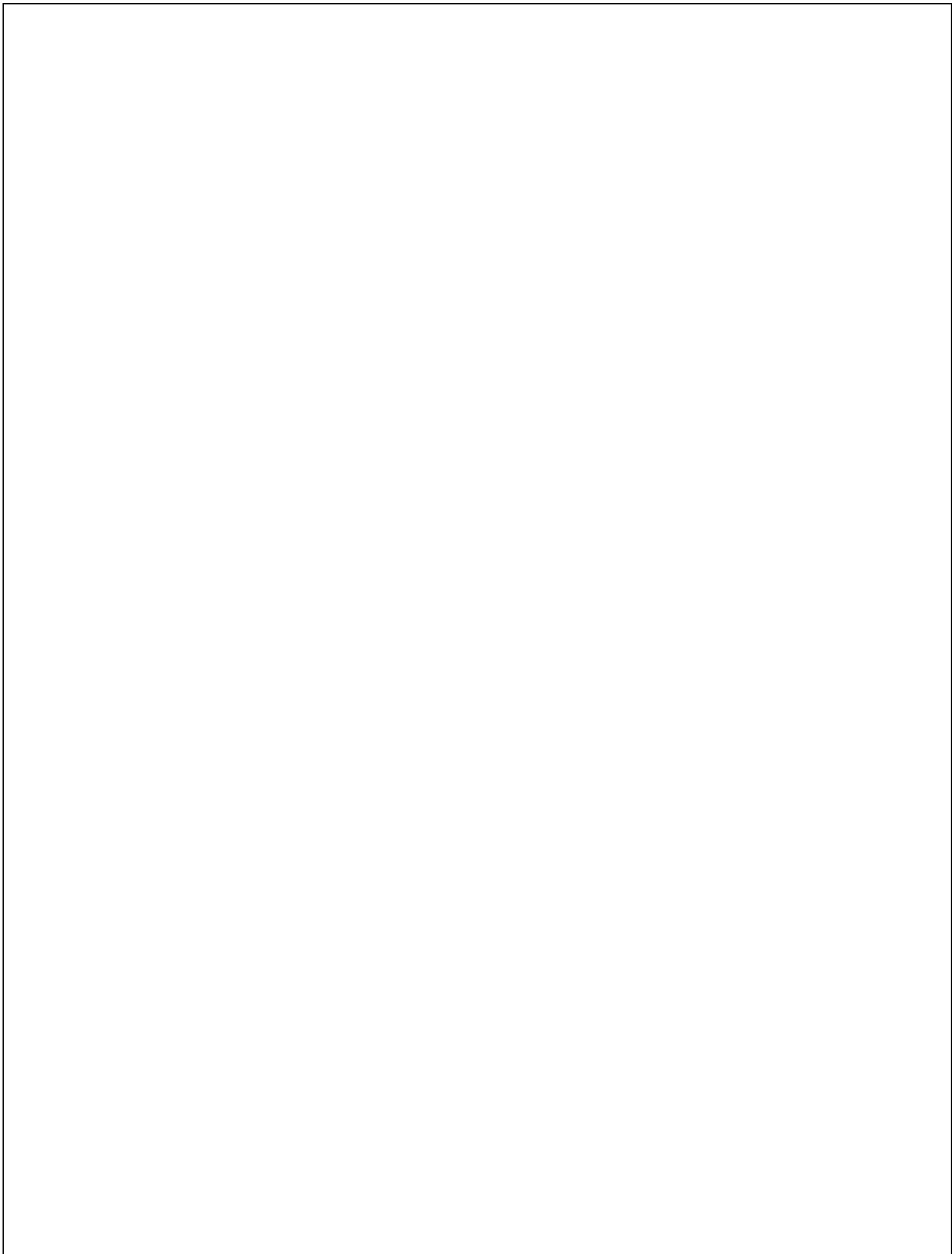
{

Ex.printStackTrace();

}

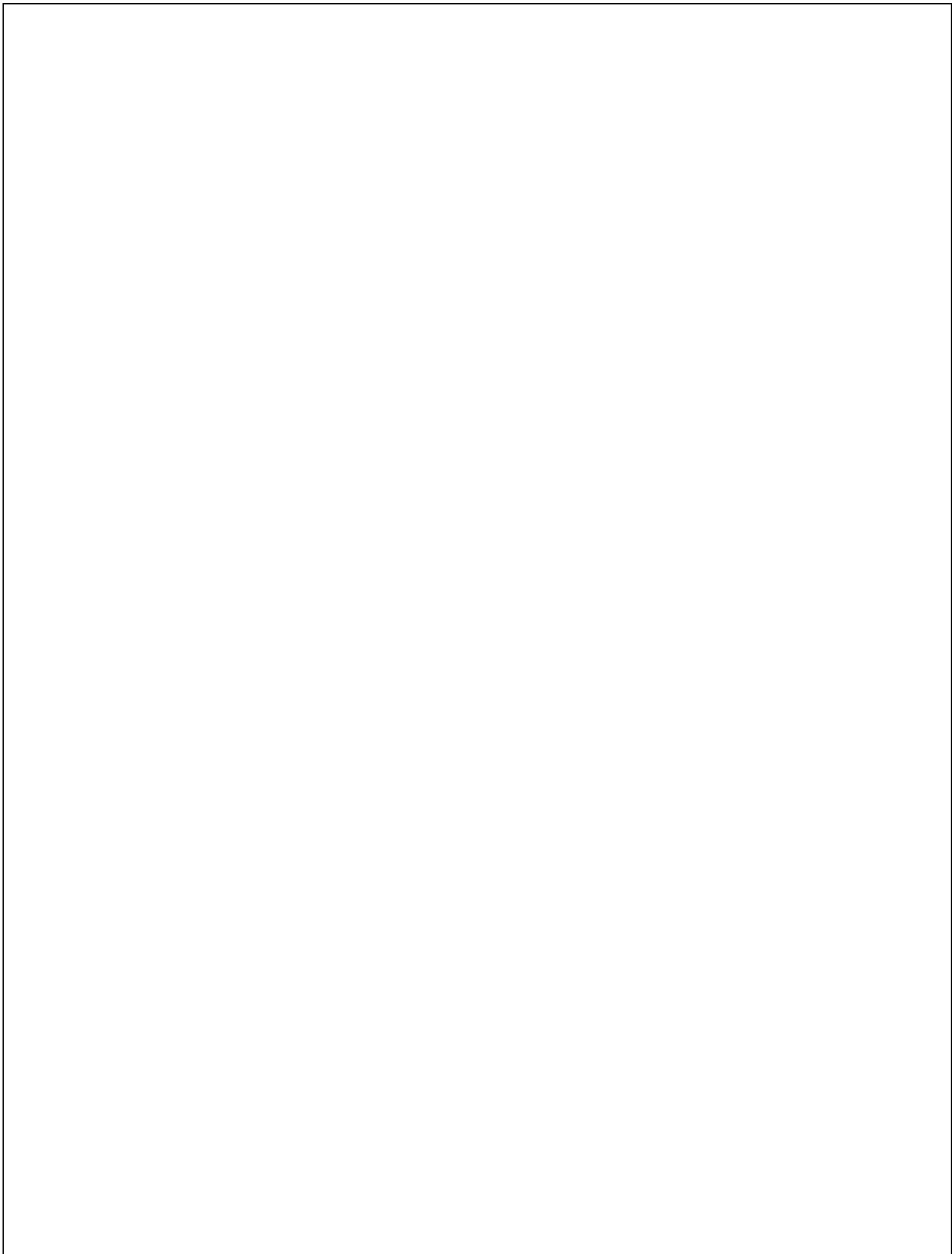
}

%>
```



Examclient.html:

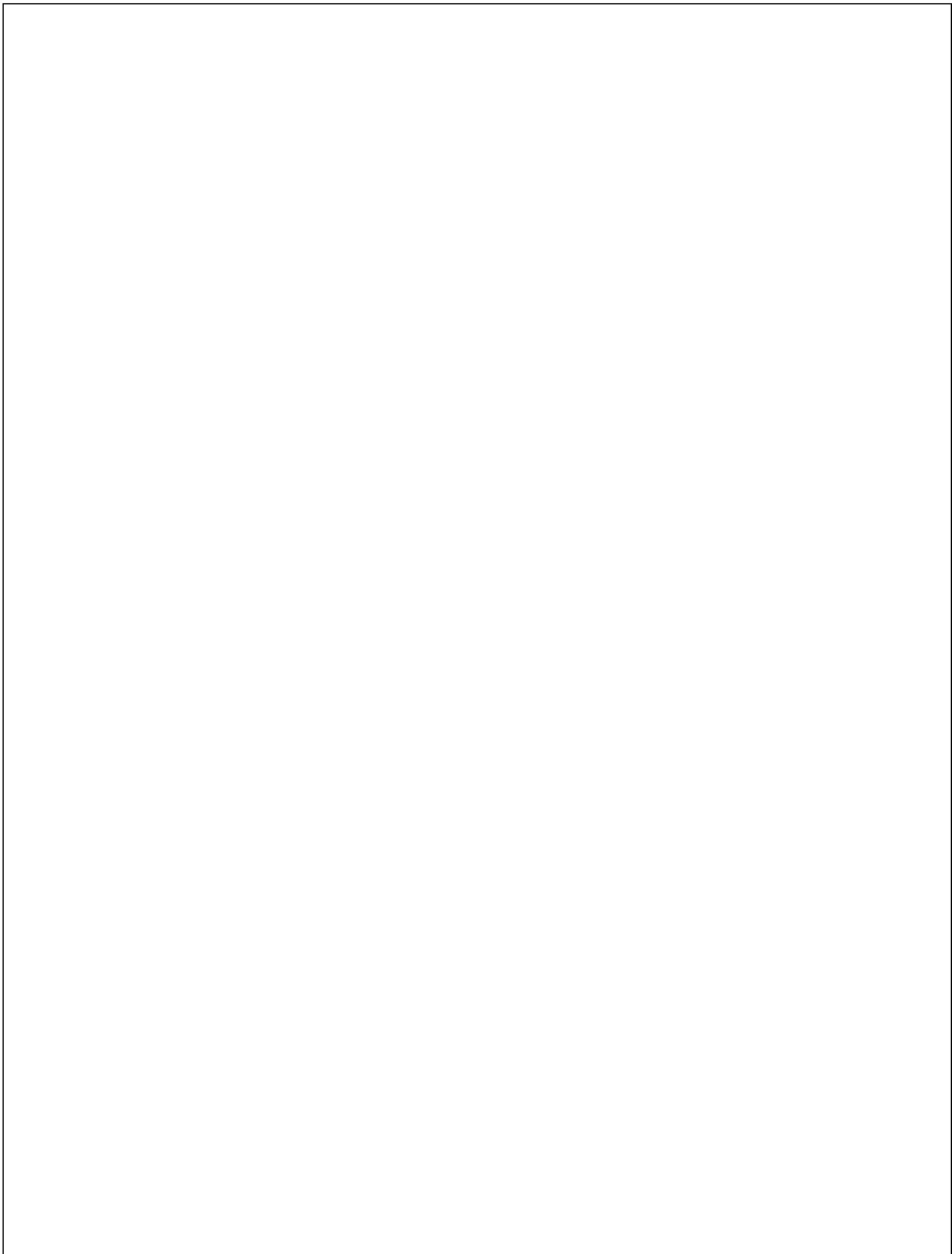
```
<html>
<head>
<title>Online Exam Client</title>
</head>
<body>
<h2 style ="text-align:center">ONLINE EXAMINATION</h2>
<h3>Answer the following question (5 marks for each correct answer)</h3>
<hr/>
<form name ="examForm" method="post" action="ExamServer.jsp">
1. Who is called as the father of computer?<br/>
<input type="radio" name="ans1" value="Sachin">Sachin
<input type="radio" name="ans1" value="Stuart">Stuart
<input type="radio" name="ans1" value="Charles Babbage">Charles Babbage
<input type="radio" name="ans1" value="Napier">Napier
<br/><br/>
2. Python was developed by?<br/>
<input type="radio" name="ans2" value="Dennis Ritchie">Dennis Ritchie
<input type="radio" name="ans2" value="Guido van Rossum"> Guido van Rossum
<input type="radio" name="ans2" value="David Ritchie">David Ritchie
<input type="radio" name="ans2" value="John">John
<br/><br/>
3. C was developed by?<br/>
<input type="radio" name="ans3" value="Dennis Ritchie">Dennis Ritchie
<input type="radio" name="ans3" value="Stroustrup">Stroustrup
```



```
<input type="radio" name="ans3" value="David Ritchie">David Ritchie  
<input type="radio" name="ans3" value="Charles Babbage">Charles Babbage  
<br/><br/>  
<input type="submit" value="Check Your Result"/>  
</form>  
</body>  
</html>
```

ExamServer.jsp:

```
<%@page contentType="text/html" language="java" import="java.sql.*"%>  
<html>  
<head>  
<title>Online Exam Server</title>  
<style type="text/css">  
body{background-color:white;font-family:courier;color:blue}  
</style>  
</head>  
<body>  
<h2 style="text-align:center">ONLINE EXAMINATION</h2>  
<p>  
<a href="examclient.html">Back To Main Page</a>  
</p>  
<hr/>  
<%  
String str1=request.getParameter("ans1");  
String str2=request.getParameter("ans2");
```



```
String str3=request.getParameter("ans3");

int mark=0;

Class.forName("org.apache.derby.jdbc.ClientDriver").newInstance();

Connection
con=DriverManager.getConnection("jdbc:derby://localhost:1527/iplab","root","root");

Statement stmt=con.createStatement();

ResultSet rs=stmt.executeQuery("SELECT * FROM examTab");

while(rs.next())

{

String dbans1=rs.getString(1);

String dbans2=rs.getString(2);

String dbans3=rs.getString(3);

if(str1.equals(dbans1))

{

mark=mark+5;

}

if(str2.equals(dbans2))

{

mark=mark+5;

}

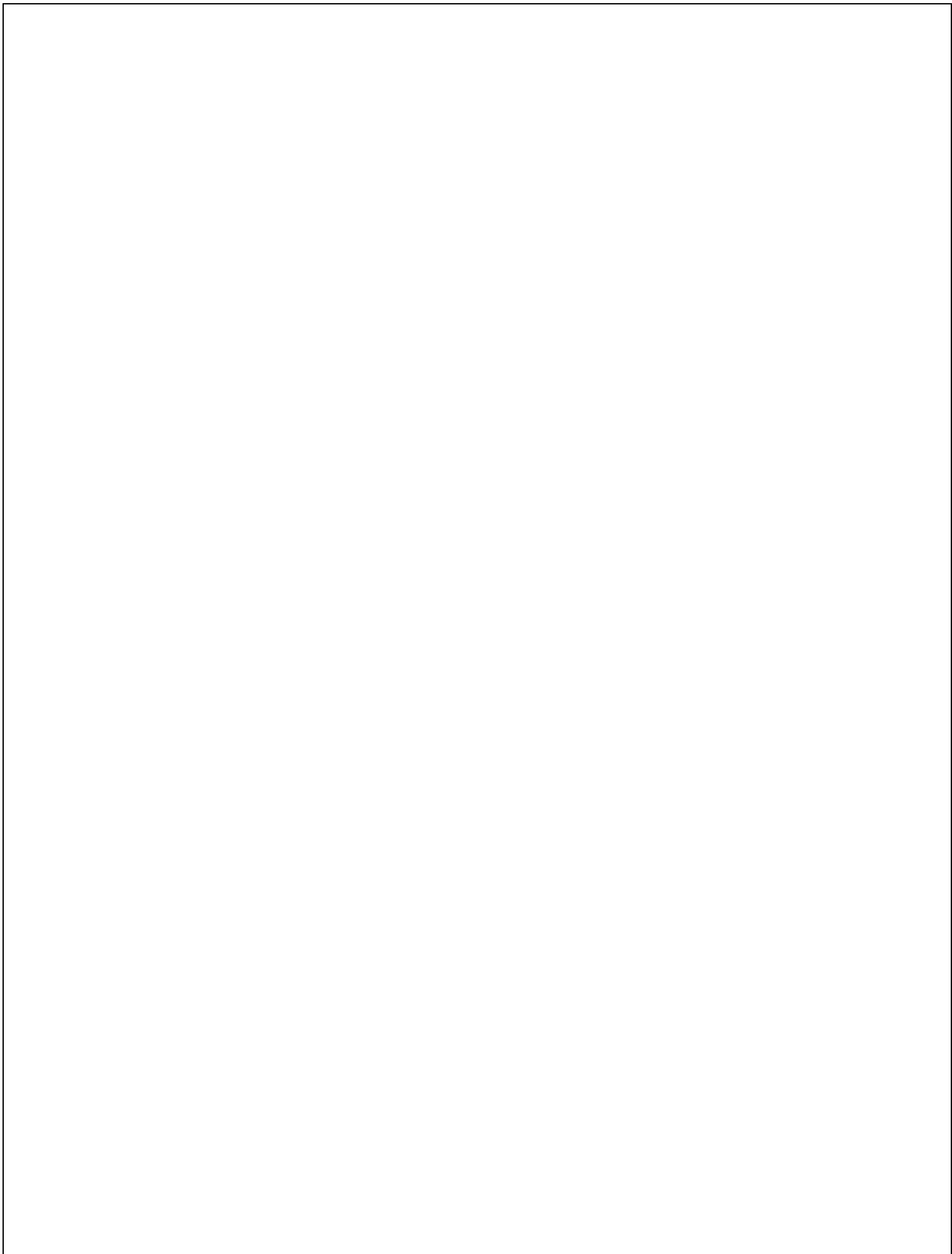
if(str3.equals(dbans3))

{

mark=mark+5;

}

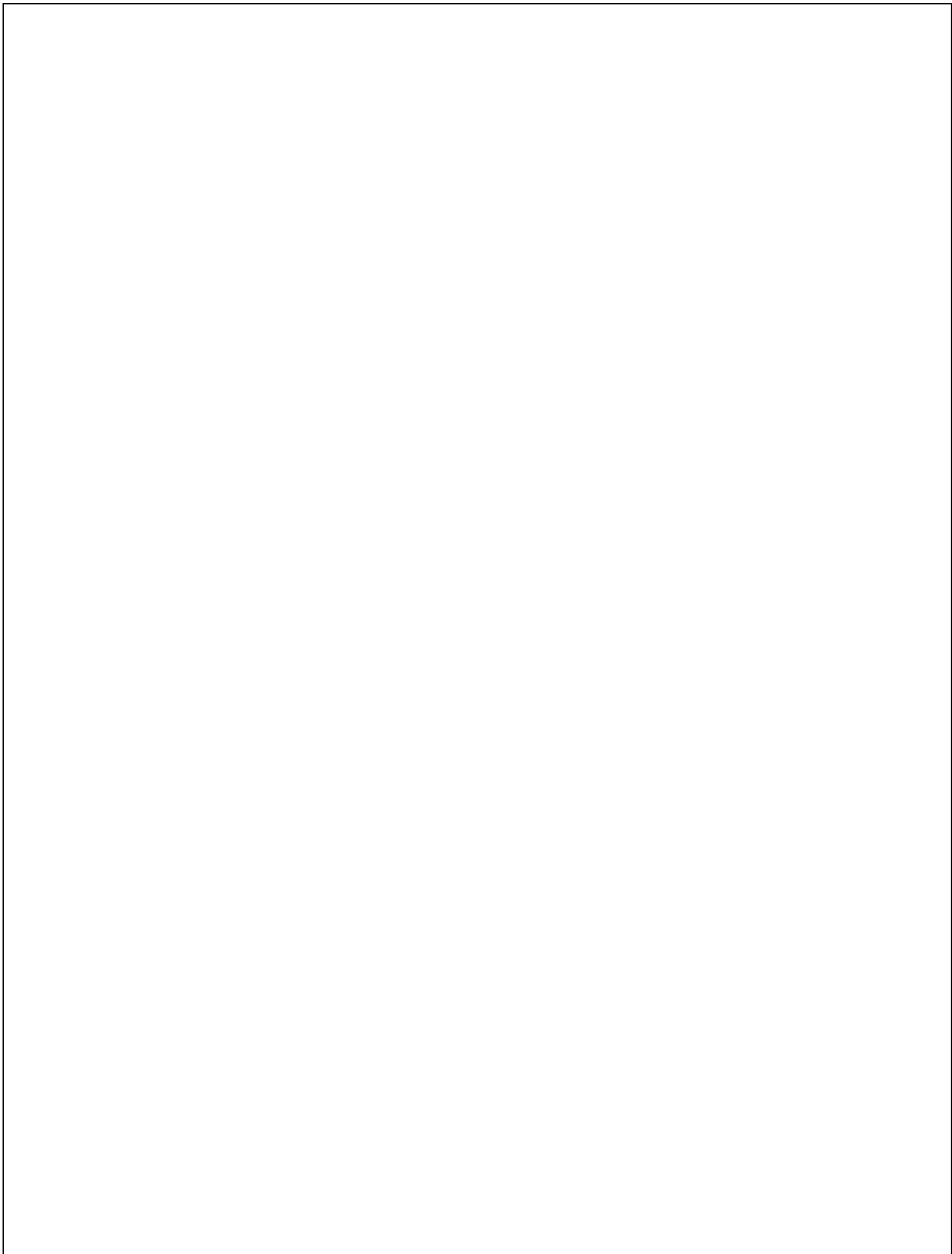
}
```



```
if(mark>=10)
{
    out.println("<h4>Your Mark Is : "+mark+"</h4>");
    out.println("<h3>Congratulations....! You Are Eligible For The Next Round...</h3>");
}
else
{
    out.println("<h4>Your Mark is : "+mark+"</h4>");
    out.println("<h3>Sorry....!! You Are Not Eligible For The Next Round...</h3>");
}
%>
</form>
</body>
</html>
```

RESULT:

Thus, the above program has been implemented and verified successfully.



Ex no: 09	PROGRAMS USING XML – SCHEMA - XSLT / XSL
Date:	

AIM:

To write a program using XML-Schema-XSLT/XSL for displaying the contents available in XML document.

ALGORITHM:

STEP 1: Start the program.

STEP 2: Create an XML document the contains data.

STEP 3: Create an XML Schema file to define the structure and rules for the XML document.

STEP 4: Link the Schema file with the XML document using the xmlns:xsi and xsi:noNamespaceSchemaLocation attributes.

STEP 5: Create an XSL or XSLT file to define how the XML data should be displayed.

STEP 6: Link the XSLT file with the XML document using:

- <?xml-stylesheet type="text/xsl" href="filename.xsl"?>

STEP 7: Open the XML file in a web browser or XML viewer.

STEP 8: Verify that the data is displayed properly according to the XSLT style.

STEP 9: End the program.

PROGRAM:

Student.XML

```
<?xml version="1.0"?>

<?xml-stylesheet type="text/xsl" version="2.0" href="student.xsl"?>

<student>

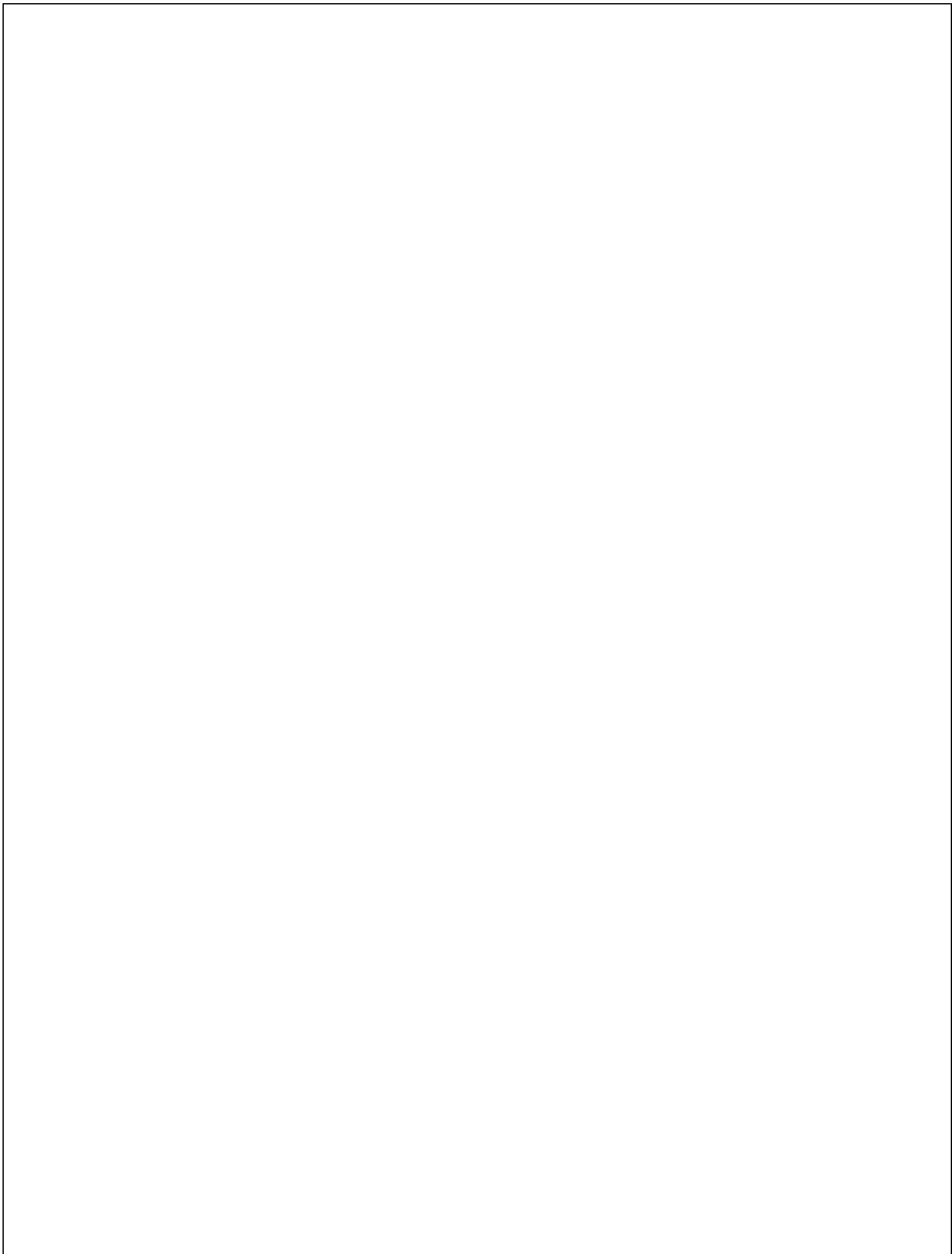
<Person-Details>

<name>Aariz</name>

<department>Mech</department>
```

OUTPUT:

```
<univ_rank>5</univ_rank>
<percentage>70 percent</percentage>
</Person-Details>
<Person-Details>
<name>Bismi</name>
<department>IT</department>
<univ_rank>4</univ_rank>
<percentage>80 percent</percentage>
</Person-Details>
<Person-Details>
<name>Sho</name>
<department>CSE</department>
<univ_rank>2</univ_rank>
<percentage>90 percent</percentage>
</Person-Details>
<Person-Details>
<name>Zayan</name>
<department>Civil</department>
<univ_rank>3</univ_rank>
<percentage>75 percent</percentage>
</Person-Details>
</student>
```



Student.XSL

```
<?xml version="1.0"?>

<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">

<html>

<body>

<h2><center>STUDENT DATABASE</center></h2>

<table align="center" border="5" cellpadding="20">

<tr bgcolor="yellow">

<th>name</th>

<th>department</th>

<th>univ_rank</th>

<th>percentage</th>

</tr>

<xsl:for-each select="student/Person-Details">

<xsl:sort select="percentage"/>

<tr bgcolor="cyan">

<td><xsl:value-of select="name"/></td>

<td><xsl:value-of select="department"/></td>

<td><xsl:value-of select="univ_rank"/></td>

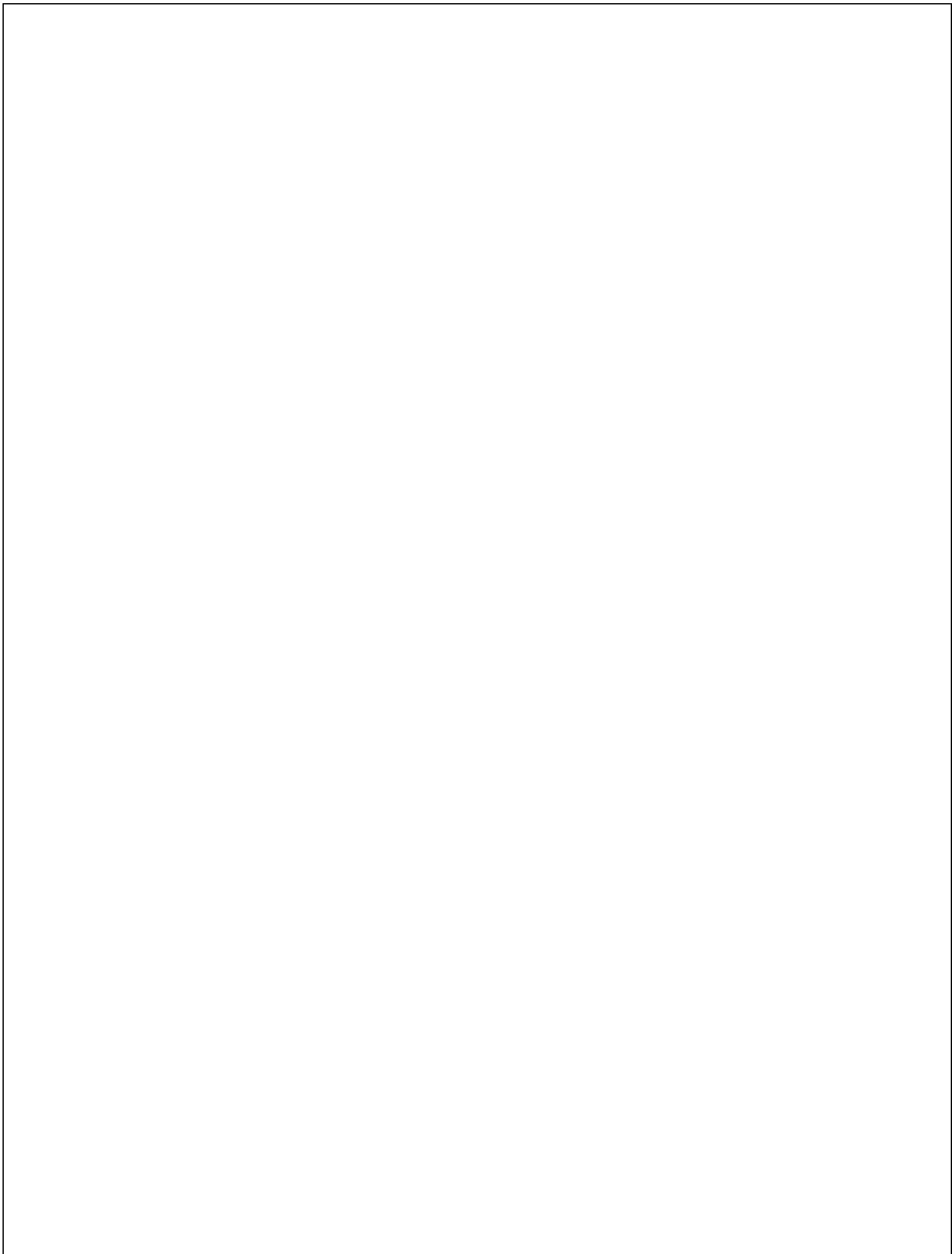
<td><xsl:value-of select="percentage"/></td>

</tr>

</xsl:for-each>

</table>

</body>
```



```
</html>  
</xsl:template>  
</xsl:stylesheet>
```

RESULT:

Thus, the above program has been implemented and verified successfully.