Week-1

1. Lists, Links and Images

1.a) Write a HTML program, to explain the working of lists.

Note: It should have an ordered list, unordered list, nested lists and ordered list in an unordered list and definition list

Aim: To create an HTML document that demonstrates the use and structure of various list types, including unordered, ordered, nested, and definition lists.

Description:

This program illustrates the fundamental HTML tags used for creating lists.

- Unordered Lists (<ul>) are used for items where the order does not matter, typically displayed with bullet points.
- Ordered Lists (<ol>) are for items where sequence is important, displayed with numbers or letters.
- Nested Lists are created by placing one list inside another list's item (<li>), allowing for hierarchical structures.
- Definition Lists (<dl>) are used to list terms (<dt>) and their corresponding descriptions (<dd>).

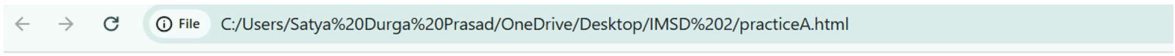
Program:

```
<!00CTYPE html>
<html lang="en"
<head>
   <meta charset="UTF-8"
   <title>HTML Lists Example</title>
</head>
<body>
   <h1>Demonstration of HTML Lists<h1>
   <!--Ordered List -->
   <h2>1. Ordered List</h2>
   <ol type="number">
      <li>First item</li>
      <li>Second item</li>
      <li>Third item</li>
   </ol>
```

```
<!--Unordered List -->
<h2>2. Unordered List</h2>
<ul>
   <li>First item</li>
   <li>Second item</li>
   <li>Third item</li>
</ul>

<!--Nested List -->
<h2>3. Nested List (Unordered inside Ordered)</h2>
<ol>
   <li>Fruits
      <ul>
          <li>Mango</li>
          <li>Grapes</li>
      </ul>
   </li>
   <li>Vegetables
      <ul>
         <li>carrot</li>
         <li>Brocolli</li>
      </ul>
   </li>
</ol>

<!--Ordered List inside Unordered List-->
<h2>4. Ordered List inside Unordered List</h2>
<ul>
   <li>steps to prepare Tea:
      <o1>
         <li>Boil Water</li>
         <li>Add tea leaves</li>
         <li>Strain and Serve</li>
      </o1>
   </li>
</ul>

 <!--Defination List-->
 <h2>5. Defination List</h2>
 <dl>
```

```
        <dt>HTML</dt>
        <dd>Hyper text markup Language,used to create webpages.</dd>
        <dt>CSS</dt>
        <dd>CasCading style sheet</dd>
    </d1>
</body>
</html>
```

Output:



HTML Lists Example

Demonstration of HTML Lists

1. Ordered List

1. First item
2. Second item
3. Third item

2. Unordered List

- First item
- Second item
- Third item

3. Nested List (Unordered inside Ordered)

1. Fruits
    - Mango
    - Grapes
2. Vegetables
    - carrot
    - Brocolli

4. Ordered List inside Unordered List

- steps to prepare Tea:
- Boil Water
- Add tea leaves
- Strain and Serve

5. Defination List

HTML
: Hyper text markup Language,used to create webpages.

CSS
: CasCading style sheet

Result:

1.b) Write a HTML program, to explain the working of hyperlinks using <a> tag and href, target Attributes.

Note: Use text to link https://www.aec.edu.in/

Use image to link https://www.aec.edu.in/?p=Gallery

Aim

To write an HTML program that demonstrates how to create hyperlinks using text and images with the anchor <a> tag, and to explain the functionality of the href and target attributes.

Description

Hyperlinks are essential for navigating between web pages. They are created using the anchor tag <a>.

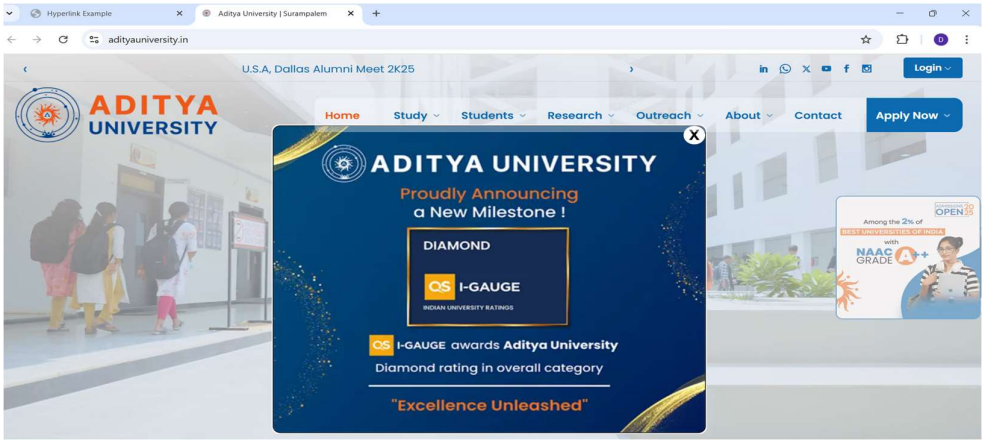
- **<a> tag:** This is the anchor tag, used to define a hyperlink. Anything between the opening <a> and closing tags becomes part of the link.
- **href attribute:** The "hypertext reference" attribute specifies the URL (Uniform Resource Locator) of the page the link goes to. This is the destination address.
- **target attribute:** This attribute specifies where to open the linked document.
    - _blank: Opens the link in a new window or tab.
    - _self: Opens the link in the same frame it was clicked in (this is the default).
    - _parent: Opens the link in the parent frame.
    - _top: Opens the link in the full body of the window.

Program:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>Hyperlink Example</title>
</head>
<body>
   <h1>Demonstrating Hyperlinks in HTML</h1>
```

```
    <!-- Text Hyperlink -->
    <h2>1. Text Hyperlink</h2>
    <p>
        Visit the
        <a href="https://adityauniversity.in/" target="_blank" rel="noopener
noreferrer">
            Aditya uiversity website
        </a>.
    </p>

    <!-- Image Hyperlink -->
    <h2>2. Image Hyperlink</h2>
    <p>
        Click the image below to visit the <strong>Gallery page</strong>:
    </p>
    <a href="https://adityauniversity.in//?p=Gallery" target="_blank"
rel="noopener noreferrer">
        <img src="C:\Users\Satya Durga Prasad\OneDrive\图片
\Screenshots\Screenshot 2025-08-31 103515.png" alt="Gallery Thumbnail"
width="300">
    </a>
</body>
</html>
```

Output


Hyperlink Example
Aditya University | Surampalem
adityauniversity.in
U.S.A, Dallas Alumni Meet 2K25
Login
ADITYA UNIVERSITY
Home
Study
Students
Research
Outreach
About
Contact
Apply Now
ADITYA UNIVERSITY
Proudly Announcing
a New Milestone !
DIAMOND
I-GAUGE
INDIAN UNIVERSITY RATINGS
I-GAUGE awards Aditya University
Diamond rating in overall category
"Excellence Unleashed"
Among the 2% of
BEST UNIVERSITIES OF INDIA
with
NAAC
GRADE A++

Result :

1.c) Create a HTML document that has your image and your friend's image with a specificheight and width. Also when clicked on the images it should navigate to their respective profiles

Aim

To create an HTML document that displays two images with specified dimensions. Each image, when clicked, will function as a hyperlink, navigating the user to a different web profile or page.

Description

This program demonstrates how to make an image a clickable link. This is achieved by nesting an <img> tag inside an <a> (anchor) tag.

- **<a> tag:** The anchor tag creates the hyperlink. Its href attribute specifies the destination URL (e.g., a social media profile).
- **<img> tag:** This tag is used to embed an image on the page.
    - src: This attribute specifies the path or URL to the image file. We will use placeholder images for this example.
    - height: This attribute sets the display height of the image in pixels.
    - width: This attribute sets the display width of the image in pixels.
    - alt: This provides alternative text, which is important for accessibility and is displayed if the image cannot be loaded.

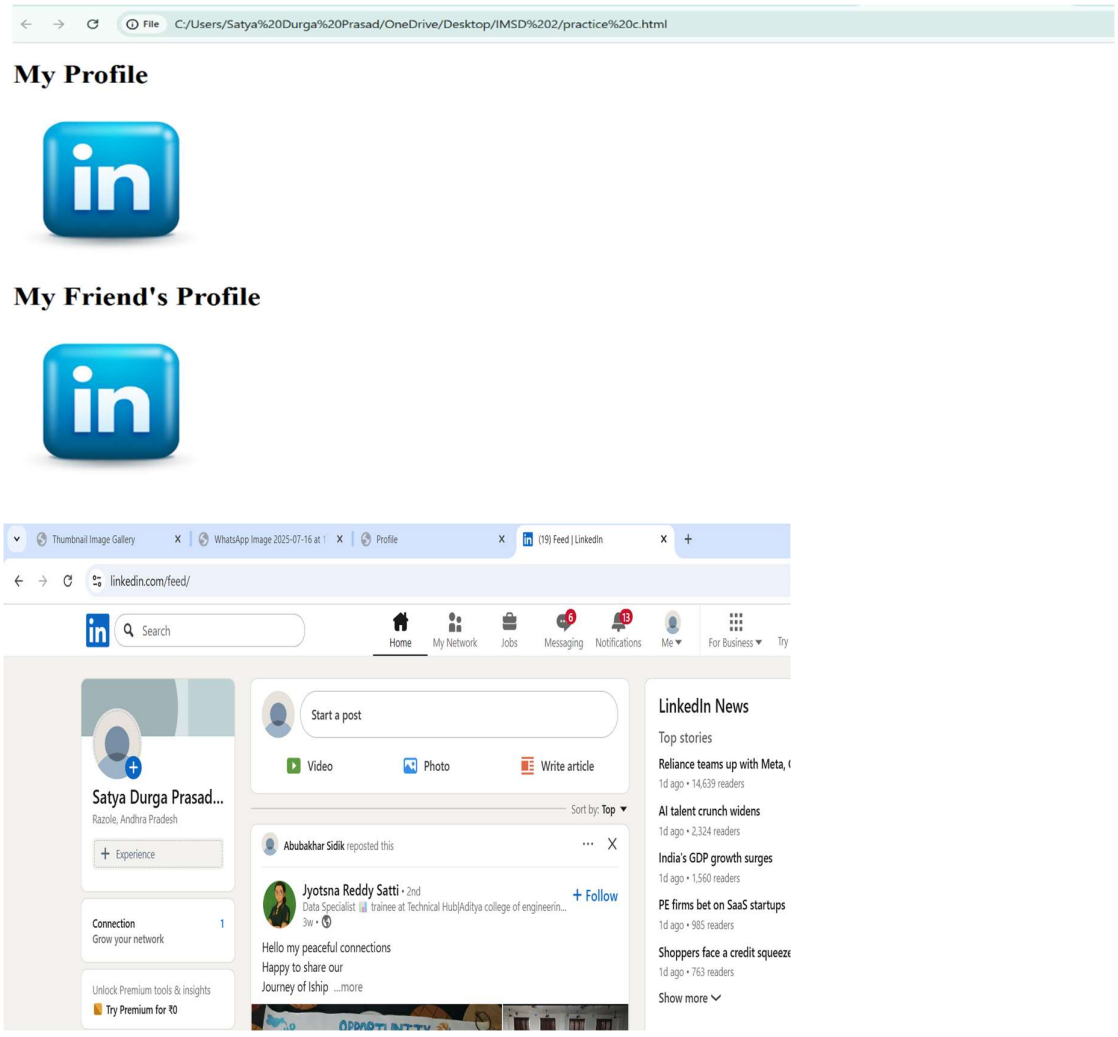
By placing the <img> tag inside the <a>... tags, the entire image becomes a clickable link.

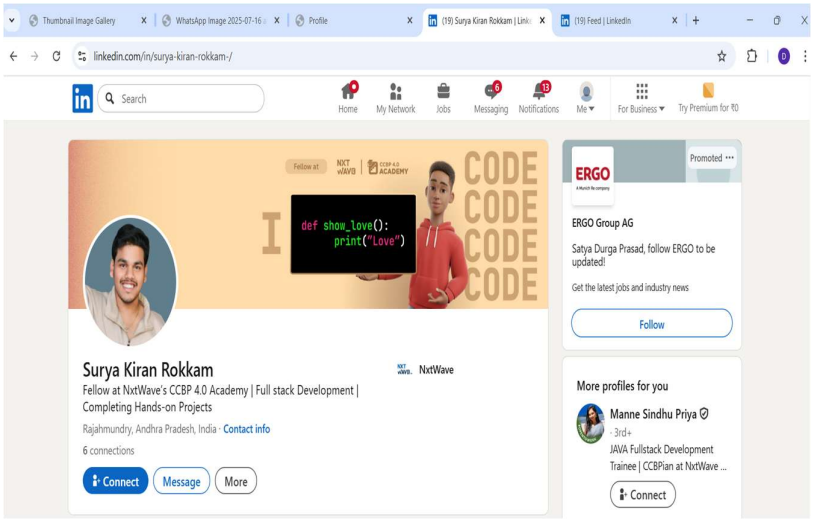
Program:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>Profile</title>
</head>
<body>
   <h1>My Profile </h1>
   <a href="https://www.linkedin.com/in/durgaprasad-pinisetty-
a4b743327?utm_source=share&utm_campaign=share_via&utm_content=profil
e&utm_medium=android_app" target="_blank">
```

```
        <img src="C:\Users\Satya Durga Prasad\OneDrive\图片\Saved Pictures\WhatsApp Image 2025-07-15 at 21.22.42_17a6a245.jpg" height="200" alt="My Profile Picture">
    </a>
    <h1>My Friend Profile</h1>
    <a href="https://www.linkedin.com/in/durga-sahith-suvvada-0770bb355?utm_source=share&utm_campaign=share_via&utm_content=profile&utm_medium=android_app" target="_blank">
        <img src="C:\Users\Satya Durga Prasad\OneDrive\图片\Saved Pictures\WhatsApp Image 2025-07-15 at 21.22.42_17a6a245.jpg" height="200" alt="Friend's Profile Picture">
    </a>
</body>
</html>
```

Output:




linkedin.com/in/surya-kiran-rokkam-/
Search
Home
My Network
Jobs
Messaging
Notifications
Me
For Business
Try Premium for ₹0
def show_love():
print("Love")
CODE
Surya Kiran Rokkam
Fellow at NxtWave's CCBP 4.0 Academy | Full stack Development | Completing Hands-on Projects
Rajahmundry, Andhra Pradesh, India · Contact info
6 connections
Connect
Message
More
NxtWave
Promoted
ERGO
ERGO Group AG
Satya Durga Prasad, follow ERGO to be updated!
Get the latest jobs and industry news
Follow
More profiles for you
Manne Sindhu Priya
· 3rd+
JAVA Fullstack Development Trainee | CCBPian at NxtWave ...
Connect

Result:

1.d). Write a HTML program, in such a way that, rather than placing large images on a page, the preferred technique is to use thumbnails by setting the height and width parameters to something like to 100*100 pixels. Each thumbnail image is also a link to a full sized version of the image. Create an image gallery using this technique

Aim

To create an HTML image gallery where small thumbnail images are displayed on the page to improve loading speed. Each thumbnail acts as a hyperlink that, when clicked, opens the full-sized version of the image.

Description

Displaying large, high-resolution images directly on a webpage can make it slow to load. A preferred technique is to use **thumbnails**—small, low-resolution versions of the images. This program demonstrates that technique.

- **Thumbnail Display:** We use the <img> tag to display the thumbnail. The src attribute points to the small image file, and we explicitly set the height and width attributes to a small size, like 100 pixels.
- **Linking to the Full Image:** Each <img> tag is wrapped inside an <a> (anchor) tag. The href attribute of this <a> tag points to the full-sized, high-resolution image file.
- **User Experience:** By setting target="_blank" on the <a> tag, the full-sized image opens in a new browser tab. This allows the user to see the large image without losing their place in the gallery.

This approach ensures the main gallery page loads quickly, and users can view the high-quality images on demand.

Program:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Thumbnail Image Gallery</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      text-align: center;
    }
```

```
        .gallery {
            display: flex;
            flex-wrap: wrap;
            justify-content: center;
            gap: 15px;
            margin-top: 20px;
        }
        .gallery a img {
            width: 100px;
            height: 100px;
            object-fit: cover;
            border: 2px solid #ccc;
            border-radius: 8px;
            transition: transform 0.3s;
        }
        .gallery a img:hover {
            transform: scale(1.1);
        }
    </style>
</head>
<body>
    <h1>My Screenshot Gallery</h1>
    <p>Click on a thumbnail to view the full-sized screenshot</p>

    <div class="gallery">
        <a href="C:\Users\Satya Durga Prasad\OneDrive\图片\Saved
Pictures\WhatsApp Image 2025-07-16 at
17.05.30_ec5de7dd.jpg"target="_blank">
            <img src="C:\Users\Satya Durga Prasad\OneDrive\图片\Saved
Pictures\WhatsApp Image 2025-07-16 at
17.05.30_ec5de7dd.jpg"alt="Screenshot 1">
        </a>
        <a href="C:\Users\Satya Durga Prasad\OneDrive\图片\Saved
Pictures\WhatsApp Image 2025-07-16 at
17.05.31_56189d4a.jpg"target="_blank">
```

```
        <img src="C:\Users\Satya Durga Prasad\OneDrive\图片\Saved
Pictures\WhatsApp Image 2025-07-16 at 17.05.31_56189d4a.jpg"
alt="Screenshot 2">
      </a>
      <a href="C:\Users\Satya Durga Prasad\OneDrive\图片\Saved
Pictures\WhatsApp Image 2025-07-16 at 17.05.32_7fc969d0.jpg"
target="_blank">
        <img src="C:\Users\Satya Durga Prasad\OneDrive\图片\Saved
Pictures\WhatsApp Image 2025-07-16 at 17.05.32_7fc969d0.jpg"
alt="Screenshot 3">
      </a>
      <a href="C:\Users\Satya Durga Prasad\OneDrive\图片\Saved
Pictures\WhatsApp Image 2025-07-16 at 17.05.32_19d3c294.jpg"
target="_blank">
        <img src="C:\Users\Satya Durga Prasad\OneDrive\图片\Saved
Pictures\WhatsApp Image 2025-07-16 at 17.05.32_19d3c294.jpg"
alt="Screenshot 4">
      </a>
   </div>
</body>
</html>
```

Output:


Thumbnail Image Gallery
WhatsApp Image 2025-07-16 at
File
C:/Users/Satya%20Durga%20Prasad/OneDrive/图片/Saved%20Pictures/WhatsApp%20Image%202025-07-16%20at%2017.05.32_19d3c294.jpg

Result :

Week-2

2.a) Write a HTML program, to explain the working of tables. (use tags: <table>, <tr>, <th>, <td> and attributes: border, rowspan, colspan)

Aim

To write an HTML program that demonstrates the creation and structure of a data table using the <table>, <tr>, <th>, and <td> tags, and illustrates how to modify the table's layout using the border, rowspan, and colspan attributes.

Description

HTML tables are used to present data in a two-dimensional grid of rows and columns. The primary tags used to create a table are:

- **<table> tag:** This is the main container for the entire table. All other table elements are nested within it.
    - **border attribute:** This attribute specifies the width of the border around the table and its cells. A value of "1" creates a thin border, making the table structure visible. (Note: In modern HTML, CSS is the preferred way to style borders).
- **<tr> tag:** This stands for "table row" and is used to define a row within the table.
- **<th> tag:** This stands for "table header." It defines a header cell in a table. Text inside a <th> tag is typically rendered as bold and centered.
- **<td> tag:** This stands for "table data." It defines a standard data cell in a table.
- **colspan attribute:** This attribute is used within a <th> or <td> tag to make a cell span (or merge) across multiple columns. For example, colspan="2" makes a cell take up the space of two columns.
- **rowspan attribute:** This attribute is used within a <th> or <td> tag to make a cell span (or merge) across multiple rows. For example, rowspan="3" makes a cell take up the space of three rows.

Program:

Rowspan:

```
<html>
<body>
<table border="1"align="center">
```

```
<tr><th rowspan="2">Name</th><td> Arjun</td></tr>
<tr><td>Harsha</td>
<tr><th rowspan="2">Mobile No</th><td>9929364279</td></tr>
<tr><td>9929364427</td>
</table>
</body>
</html>
```

Columspan

```
<html>
<body>
<table border="1"align="center">
<tr>
<th colspan="2">phone number</th>
<th colspan="2"> email id</th>
</tr>
<tr>
<th> phone number 1</th>
<th> phone number 2</th>
<th> email id 1</th>
<th> email id 2</th>
</tr>
<tr>
<td>8888999999</td>
<td>7777883434</td>
<td> josha@gmail.com</td>
<td>harsha@gmail.com</td>
</tr>
</table>
</body>
</html>
```

Output For Rowspan:

Name	Arjun
	Harsha
Mobile No	9929364279
	9929364427

Output for coloumspan:

phone number		email id	
phone number 1	phone number 2	email id 1	email id 2
8888999999	7777883434	josha@gmail.com	harsha@gmail.com

Result:

2.b) Write a HTML program, to explain the working of tables by preparing a timetable. (Note: Use <caption> tag to set the caption to the table & also use cell spacing, cell padding, border, rowspan, colspan etc.).

Aim:

To create a visually structured class timetable using an HTML table, demonstrating the practical application of various table tags and attributes like `<caption>`, `rowspan`, `colspan`, `border`, `cellspacing`, and `cellpadding`.

Description:

This program constructs an HTML table to display a weekly class schedule. The `<table>` element is the foundation. The `border="2"` attribute creates a visible border around the table and its cells. `cellspacing="5"` adds space between individual cells, and `cellpadding="15"` adds padding inside each cell, making the content more readable. The `<caption>` tag provides a title for the table. We use `<th>` for table headers (like days and time slots) and `<td>` for the data cells. The `colspan` attribute is used to merge several columns for the "Lunch" break, indicating it spans across all periods. The `rowspan` attribute is used to merge rows, perfect for a lab session that covers multiple periods.

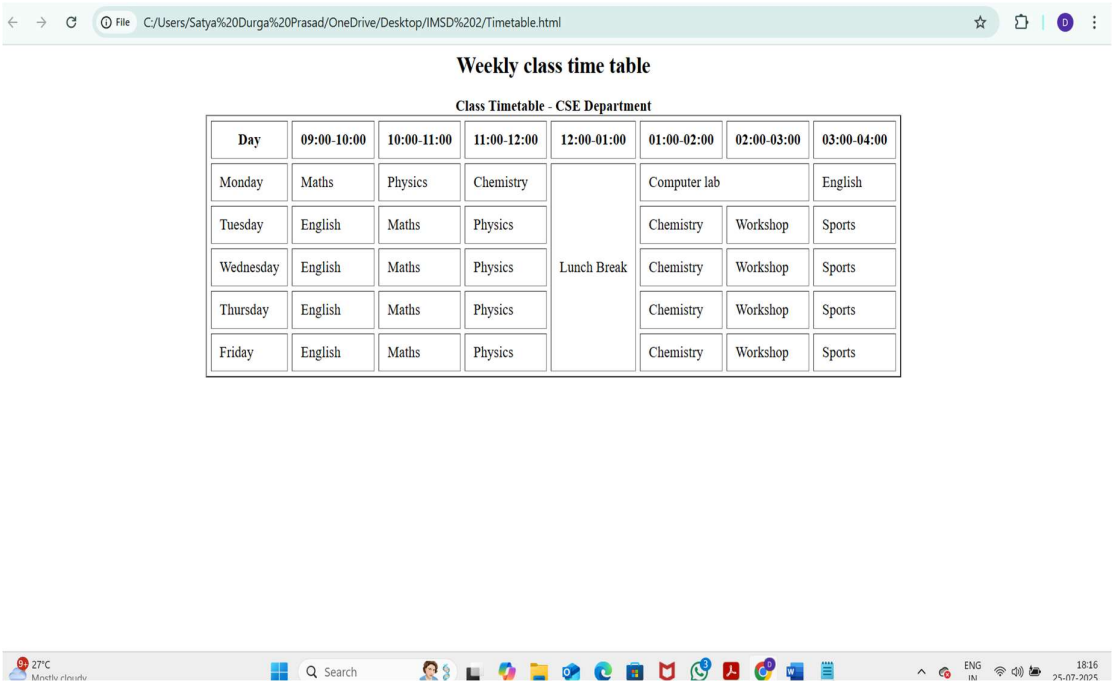
Program:

```
<html>
<head>
<title>Time table</title>
</head>
<body>
<h2 style="text-align:center;">Weekly class time table</h2>
<table align="center" border="2" cellspacing="5" cellpadding="10">
<caption><strong>Class Timetable - CSE Department</strong></caption>
<tr>
<th>Day</th>
<th>09:00-10:00</th>
<th>10:00-11:00</th>
<th>11:00-12:00</th>
<th>12:00-01:00</th>
<th>01:00-02:00</th>
<th>02:00-03:00</th>
<th>03:00-04:00</th>
```

```
</tr>
<tr>
<td>Monday</td>
<td>Maths</td>
<td>Physics</td>
<td>Chemistry</td>
<td rowspan="5" style="text-align:center;">Lunch Break</td>
<td colspan="2">Computer lab</td>
<td>English</td>
</tr>
<tr>
<td>Tuesday</td>
<td>English</td>
<td>Maths</td>
<td>Physics</td>
<td>Chemistry</td>
<td>Workshop</td>
<td>Sports</td>
</tr>
<tr>
<td>Wednesday</td>
<td>English</td>
<td>Maths</td>
<td>Physics</td>
<td>Chemistry</td>
<td>Workshop</td>
<td>Sports</td>
</tr>
<tr>
<td>Thursday</td>
<td>English</td>
<td>Maths</td>
<td>Physics</td>
<td>Chemistry</td>
<td>Workshop</td>
<td>Sports</td>
</tr>
<tr>
<td>Friday</td>
<td>English</td>
```

```
<td>Maths</td>
<td>Physics</td>
<td>Chemistry</td>
<td>Workshop</td>
<td>Sports</td>
</tr>
</table>
</body>
</html>
```

Output:



Result :

2.c) Write a HTML program, to explain the working of forms by designing Registration form.

(Note: Include text field, password field, number field, date of birth field, checkboxes, radio buttons, list boxes using <select>&<option> tags, <text area> and two buttons ie: submit and reset. Use tables to provide a better view).

Aim:

To design a user registration form using HTML to demonstrate the functionality of various form elements. The form will be structured using a table for a clean layout and will include input fields for text, passwords, numbers, dates, checkboxes, radio buttons, a selection list, a text area, and action buttons.

Description:

This program utilizes the `<form>` tag as a container for all input elements. A `<table>` is used to align the labels and their corresponding input fields neatly in rows and columns. The form includes various input types: `type="text"` for names, `type="password"` to obscure characters, `type="number"` for age, and `type="date"` for date of birth. It also demonstrates `type="radio"` for exclusive choices (gender), `type="checkbox"` for multiple selections (hobbies), the `<select>` and `<option>` tags for a dropdown list (country), and a `<textarea>` for longer text input (address). Finally, `<input type="submit">` and `<input type="reset">` buttons are provided to submit the form data and clear all entries, respectively.

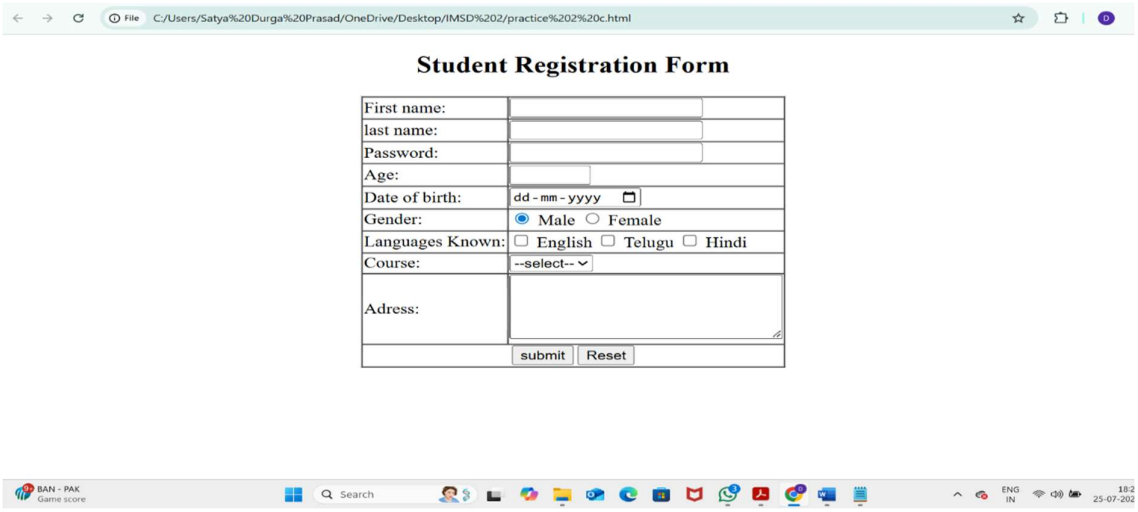
Program:

```
<!DOCTYPE html>
<html>
<head>
    <title>Registration Form</title>
</head>
<body bgcolor="white">
     <h2 align="center">Student Registration Form</h2>
     <form action="#" method="post">
      <table border="1" align="center" cellpading="10" cellspacing="0">
      <tr>
         <td><label for="fname">First name:</label></td>
         <td><input type="text" id="fname" name="fname" required></td>
```

```
    </tr>
     <tr>
       <td><label for="lname">last name:</label></td>
       <td><input type="text" id="lname" name="lname" required></td>
    </tr>
     <tr>
       <td><label for="Password">Password:</label></td>
       <td><input type="Password" id="Password" name="Password"
required></td>
    </tr>
     <tr>
       <td><label for="age">Age:</label></td>
       <td><input type="number" id="age" name="age" min="0"
max="100"></td>
    </tr>
     <tr>
       <td><label for="dob">Date of birth:</label></td>
       <td><input type="date" id="dob" name="dob"></td>
    </tr>
     <tr>
       <td>Gender:</td>
       <td>
          <input type="radio" id="male" name="gender" value="Male">
          <label for="male">Male</label>
          <input type="radio" id="female" name="gender" value="Female">
          <label for="female">Female</label>
     </td>
     </tr>
     <tr>
       <td>Languages Known:</td>
       <td>
          <input type="checkbox" id="english" name="Language"
value="English">
          <label for="english">English</label>

          <input type="checkbox" id="telugu" name="Language"
value="Telugu">
          <label for="telugu">Telugu</label>
```

```
            <input type="checkbox" id="hindi" name="Language"
value="Hindi">
             <label for="hindi">Hindi</label>
        </td>
        </tr>
        <tr>
         <td><label for="course">Course:</label></td>
         <td>
             <select id="course" name="course">
                <option value="">--select--</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>
           </td>
           </tr>
           <tr>
             <td><label for="address">Adress:</label></td>
             <td><textarea id ="address" name="address" rows="4"
cols="30"></textarea></td>
           </tr>
           <tr>
              <td colspan="2" align="center">
                  <input type="submit" value="submit">
                  <input type="reset" value="Reset">
             </td>
          </tr>
         </table>
        </form>
</body>
</html>
```

Output:



Result :

2.d) Write a HTML program, to explain the working of frames, such that page is to be divided into 3 parts on either direction. (Note: first frame image, second frame paragraph, third frame hyperlink and also make sure using no frame attribute such that frames to be fixed

Aim

To demonstrate the working of HTML frames by creating a webpage divided into three fixed, non-resizable sections using the <frameset> tag. Each section will load a separate HTML document: one with an image, one with a paragraph, and one with a hyperlink.

Description

This program uses the **<frameset>** element to partition the browser window into multiple sections, or "frames." The main HTML document does not use a <body> tag; instead, it uses <frameset> to define the layout. In this example, rows="25%, 50%, *" divides the page into three horizontal rows. The first row takes up 25% of the height, the second takes up 50%, and the third (*) takes up the remaining space.

Each **<frame>** tag specifies a source (src) document to load into that section. The **noresize** attribute is added to each frame to prevent the user from resizing the divisions.

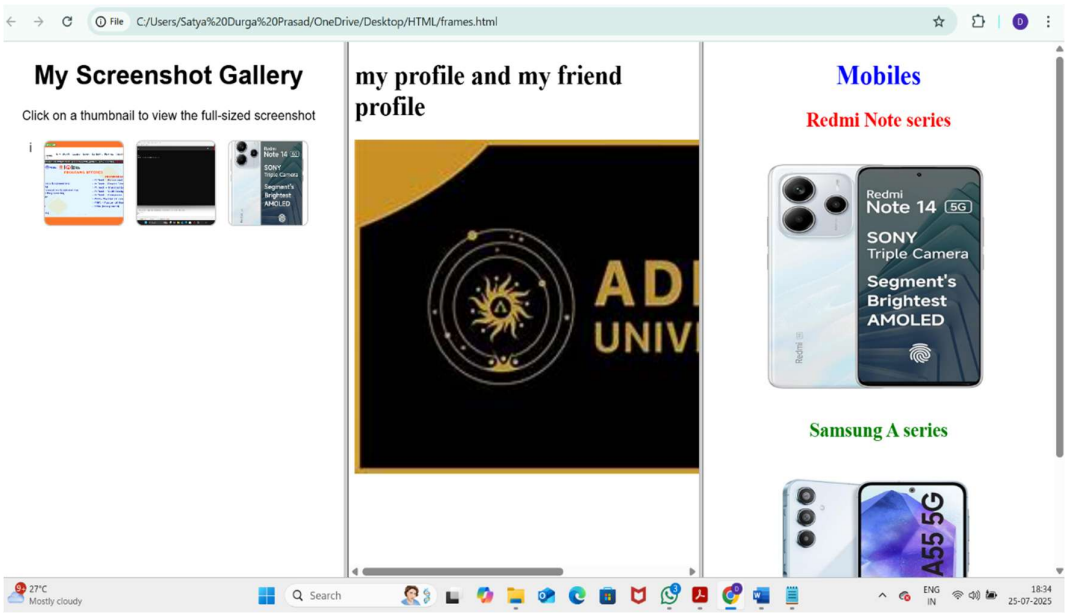
Note: The <frameset> and <frame> tags are **obsolete in HTML5** and are no longer recommended for modern web development. They have been replaced by more flexible and powerful CSS layout techniques (like Flexbox and Grid) and <iframe> elements for embedding content. This example is for educational purposes to understand a legacy feature of HTML.

Program:

```
<!DOCTYPE html>
<html>
<head>
    <title>Frames Example - Verticalsplit</title>
</head>
<frameset cols="33%,33%,34%">
    <frame src="thumbnail.html" name="thumbFrame">
    <frame src="profile.html" name="profileFrame">
    <frame src="Mobile.html" name="mobileFrame">
    <noframes>
```

```
        <body>
            <p>Your browser does not support frames. Please update your
browser.</p>
        </body>
    </noframes>
</framset>
</html>
```

Output:



Result :

Week-3

3.a) Write a HTML program, that makes use of <article>, <aside>, <figure>, <figcaption>, <footer>, <header>, <main>, <nav>, <section>, , tags.

Aim: To design a webpage using HTML5 semantic tags such as <article>, <aside>, <figure>, <figcaption>, <footer>, <header>, <main>, <nav>, <section>, , and to demonstrate the structure and meaning of each element.

Description:

This program creates a travel blog webpage that uses HTML5 semantic tags for better structure and readability. The <header> tag defines the page heading, while the <nav> tag contains the navigation menu for moving to different sections. The <main> tag holds the central content, and inside it, <article> is used to represent a self-contained story. The <section> tag groups related content inside the article, and <figure> with <figcaption> is used to insert an image along with a descriptive caption. The <aside> tag contains tips related to the main topic, and <footer> displays copyright details. The tag is a generic container, while is used for inline styling or grouping text inside other elements.

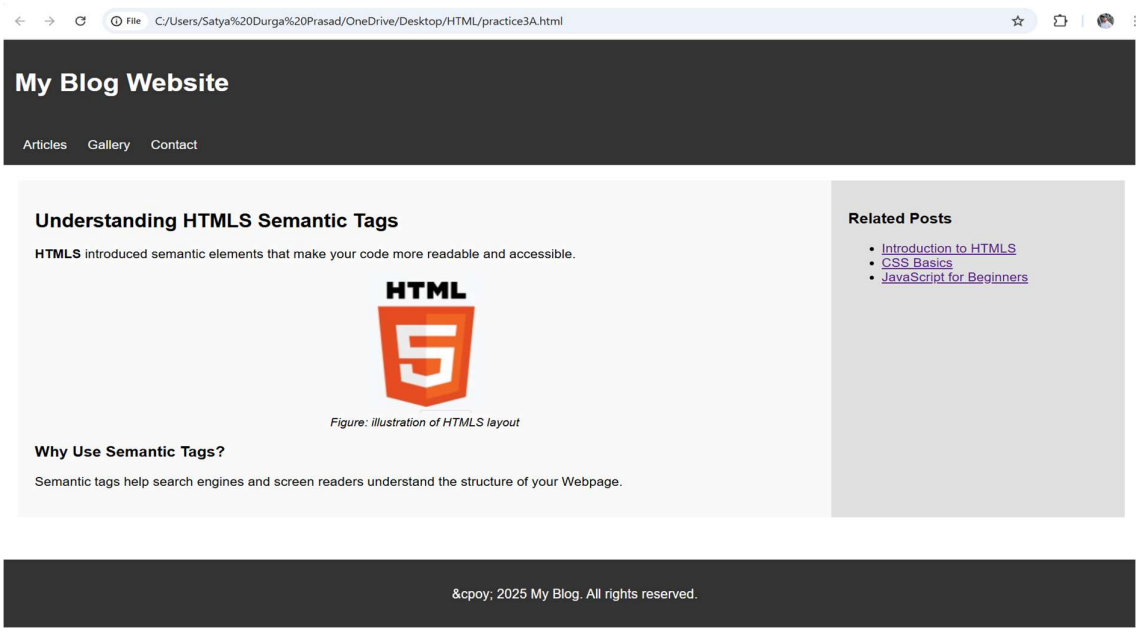
Program:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>HTMLS semantic Elements Demo</title>
    <style>
      body {
          font-family:Arial, sans-serif;
          margin: 0;
      }
      header,nav,footer {
           background-color:#333;
           color: White;
           padding:1em;
      }
      nav a {
         color:White;
```

```
        margin: 0 10px;
        text-decoration: none;
    }
    main {
       display: flex;
       padding:20px;
     }
     article {
         flex: 3;
         padding: 20px;
         background-color: #f9f9f9;
    }
    aside {
        flex: 1;
        padding: 20px;
        background-color: #e0e0e0;
     }
     figure {
         margin: 0;
         text-align: center;
     }
     figcaption {
           font-style: italic;
           font-size: 0.9em;
     }
     footer {
          text-align: center;
     }
    </style>
</head>
<body>

<header>
   <h1>My Blog Website</h1>
</header>
<nav>
    <a href="home.html>Home</a>
    <a href="">Articles</a>
    <a href="#">Gallery</a>
    <a href="#">Contact</a>
```

```
</nav>

<main>
    <article>
        <h2>Understanding HTMLS Semantic Tags</h2>
        <p><span style="font-weight:bold;">HTMLS</span> introduced
semantic elements that make your code more readable and accessible.</p>
       <figure>
         <img src="C:\Users\Satya Durga Prasad\OneDrive\文档\WhatsApp
Image 2025-07-30 at 15.09.53_3903ea76.jpg " alt="HTMLS illustration">
         <figcaption>Figure: illustration of HTMLS layout</figcaption>
        </figure>
       <section>
           <h3>Why Use Semantic Tags?</h3>
           <p>Semantic tags help search engines and screen readers understand
the structure of your Webpage.</p>
      </section>
   </article>

  <aside>
      <h3>Related Posts</h3>
      <ul>
         <li><a href="#">Introduction to HTMLS</a></li>
         <li><a href="#">CSS Basics</a></li>
         <li><a href="#">JavaScript for Beginners</a></li>
      </ul>
    </aside>
</main>

<footer>
     <p>&cpoy; 2025 My Blog. All rights reserved.</p>
</footer>

</head>
</html>
```

Output:



Result:

3.b) Write a HTML program, to embed audio and video into HTML web page.

Aim:

To create an HTML5 webpage that embeds audio and video files using <audio> and <video> tags.

Description:

This program demonstrates multimedia embedding in HTML5. The <audio> tag is used to insert an audio file (nature.mp3) with controls that allow the user to play, pause, and adjust the volume. The <video> tag is used to embed a video file (nature.mp4) with similar controls and specified width and height. The <source> element inside both tags specifies the file path and format, making it possible for the browser to choose the correct media file. Alternative text is provided for browsers that do not support these tags.

Program:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>Audio and Video Embedding</title>
</head>
<body>
   <h1>Embedding Audio and Video in HTML</h1>

   <!-- Audio Section -->
   <h2>Sample Audio-1</h2>
   <audio controls>
      <source src="C:\Users\Satya Durga Prasad\OneDrive\Music\WhatsApp
Audio 2025-07-30 at 15.50.42_2f2c85ac.mp3" type="audio/mpeg">
      Your browser does not support the audio element.
   </audio>

   <h2>Sample Audio-2</h2>
   <audio controls>
      <source src="C:\Users\Satya Durga Prasad\OneDrive\Music\WhatsApp
Audio 2025-07-30 at 15.50.43_16334670.mp3" type="audio/mpeg">
      Your browser does not support the audio element.
   </audio>
```

```
    <!-- Video Section -->
    <h2>Sample Video-1</h2>
    <video width="640" height="360" controls>
        <source src="C:\Users\Satya Durga Prasad\OneDrive\Videos\WhatsApp
Video 2025-07-31 at 20.27.46_d4e52420.mp4" type="video/mp4">
        Your browser does not support the video tag.
    </video>

    <h2>Sample Video-2</h2>
    <video width="640" height="360" controls>
        <source src="C:\Users\Satya Durga Prasad\OneDrive\Videos\WhatsApp
Video 2025-07-31 at 20.27.46_460c622f.mp4"  type="video/mp4">
        Your browser does not support the video tag.
    </video>
</body>
</html>
```

Output:

Embedding Audio and Video in HTML

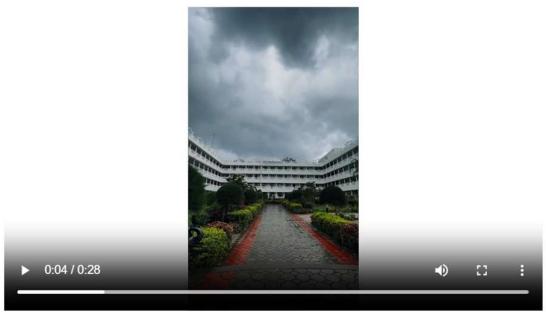
Sample Audio-1

0:12 / 0:31

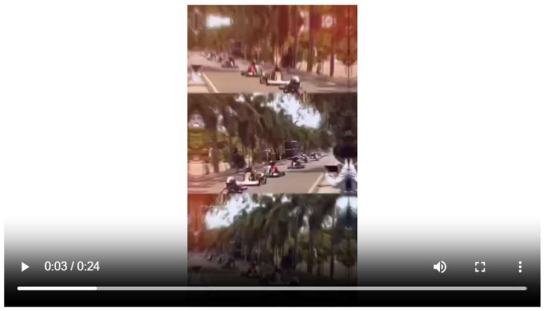
Sample Audio-2

0:04 / 0:30

Sample Video-1



Sample Video-2



Result:

3.c) Write a program to apply different types (or levels of styles or style specification formats) -inline, internal, external styles to HTML elements. (identify selector, property and value).

Aim:

To apply CSS styles to HTML elements using different types of style specifications: inline, internal, and external CSS.

Description:

The program uses three CSS styling methods:

External CSS – A separate styles.css file contains rules for headings (h1) to make the text dark blue and center-aligned. It is linked to the HTML file using the <link> tag.

Internal CSS – CSS rules are placed inside the <style> tag within the HTML head section to set paragraph text color to green and define a class .highlight that gives text a yellow background.

Inline CSS – The style attribute is used inside an HTML tag to directly apply styles such as red text color and bold font to a specific paragraph.

Program:

External css:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
   <h1>This is a heading</h1>
   <p>This is a paragraph</p>
</body>
</html>
```

Internal css:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>INLINE STYLESHEET</title>
```

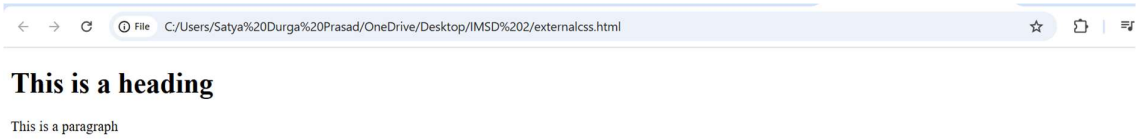
```
    <style>
        body
        {
            background-color:coral;
        }
    </style>
</head>
<body>
    <h1>This is a heading</h1>
    <p>This is a paragraph</p>
</body>
</html>
</html>
```

Iniline css:

```
<html>
<body>
<h1 style="color:green;text-align:center;">This is a heading</h1>
<p style="color:orange;">This is a paragraph.</p>
</body>
</html>
```

Output:

External css:



Internal css:

Iniline css:

Result:

Week 4

4.a Write a program to apply different types of selector forms

i). Simple selector (element, id, class, group, universal)

Aim:

To demonstrate the use of various CSS selectors—element, ID, class, group, and universal—by applying distinct styles to HTML elements.

Description:

This HTML document showcases five types of CSS selectors:

1. Element Selector

- Syntax: p { color: blue; }
- Effect: All

elements are styled with blue text.

2. ID Selector

- Syntax: #heading { color: green; }
- Effect: The

element with id="heading" is styled with green text.

3. Class Selector

- Syntax: .highlight { background-color: yellow; }
- Effect: Any element with class="highlight" gets a yellow background.

4. Group Selector

- Syntax: h2, h3 { color: red; }
- Effect: Both

and

elements are styled with red text.

5. Universal Selector

- Syntax: * { font-family: Arial; }
- Effect: All elements in the document use the Arial font.

Let me know if you'd like a quiz or diagram to reinforce this!

Program:

<!DOCTYPE html>

```
<html>
<head>
   <title>Simple Selectors</title>
   <style>
      /* Element selector */
      p { color: blue; }

      /* ID selector */
      #heading { color: green; }

      /* Class selector */
      .highlight { background-color: yellow; }

      /* Group selector */
      h2, h3 { color: red; }

      /* Universal selector */
      * { font-family: Arial; }
   </style>
</head>
<body>
   <h1 id="heading">This is a heading with ID selector</h1>
   <p>This is styled using element selector</p>
   <p class="highlight">This paragraph uses class selector</p>
   <h2>Grouped selector applied here</h2>
   <h3>Grouped selector applied here too</h3>
</body>
</html>
```

Output:

Result:

4.a.ii) Combinator selector (descendant, child, adjacent sibling, general sibling)

Aim

To demonstrate the use of CSS combinator selectors—which define relationships between elements—through a practical HTML and CSS example.

Description

This program showcases four types of CSS combinator selectors:

1. **Descendant Selector (div p)**
    - Targets all <p> elements nested anywhere inside a .
    - Example: Paragraph inside a <section> within is styled blue.
2. **Child Selector (div > p)**
    - Targets only direct child <p> elements of .
    - Example: First <p> inside gets a yellow background**.**
3. **Adjacent Sibling Selector (h2 + p)**
    - Targets the first <p> immediately following an <h2>.
    - Example: Styled red, bold, and italic**.**
4. **General Sibling Selector (h2 ~ p)**
    - Targets all <p> elements that follow an <h2>, regardless of position.
    - Example: All subsequent <p>s are italicized.

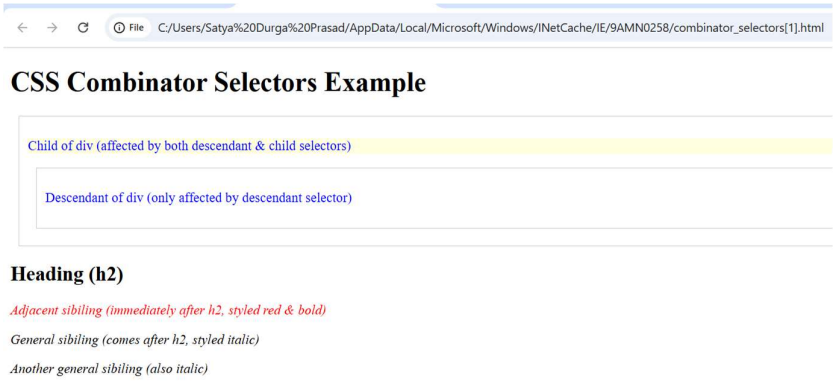
The layout includes a heading, a styled container with nested elements, and a sequence of paragraphs to illustrate sibling relationships. Each selector is visually differentiated using color, font style, and background to make the relationships clear**.**

Program:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>CSS Combinator Selectors</title>
 <style>
  /* Descendant Selector (div p) */
  div p{
    color: blue;
  }
```

```
  /* Child Selector (div > p) */
  div > p{
    background-color: lightyellow;
  }

  /* Adjacent sibling Selector (h2 + p) */
  h2 + p {
    color: red;
    font-weight: bold;
  }

  /* General Sibiling selector (h2 ~ p) */
  h2 ~ p{
    font-style: italic;
  }

  /* Styling for visibility */
  div,section{
   border:1px solid #ccc;
   padding:10px;
   margin:10px;
   }
  </style>
</head>
<body>

  <h1>CSS Combinator Selectors Example</h1>
  <div>
   <p>Child of div (affected by both descendant & child selectors)</p>
   <section>
    <p>Descendant of div (only affected by descendant selector)</p>
   </section>
  </div>

  <h2>Heading (h2)</h2>
  <p>Adjacent sibiling (immediately after h2, styled red & bold)</p>
  <p>General sibiling (comes after h2, styled italic)</p>
  <p>Another general sibiling (also italic)</p>
</body>
</html>
```

Output:



Result:

4.a.iii) Pseudo-class selector

Aim:

To demonstrate the use of CSS pseudo-class selectors that style elements based on their state, position, or user interaction.

Description :

This program demonstrates the use of **CSS pseudo-class selectors** to style HTML elements based on their dynamic states and structural positions:

- **Link pseudo-classes** (:link, :visited, :hover, :active) are applied to anchor tags to change their color depending on whether the link is unvisited, visited, hovered over, or actively clicked.
- **Focus pseudo-class** (:focus) is used to style an input field when it gains focus, changing its border and background color to visually indicate user interaction.
- **List item pseudo-classes** (:first-child, :last-child, :nth-child(n)) are used to style specific
- elements within an unordered list based on their position. The first item is colored gold, the second pink, and the last blue.
- **Checked pseudo-class** (:checked) is used with a checkbox input. When the checkbox is selected, the adjacent label is styled with green color and bold font to reflect the checked state.

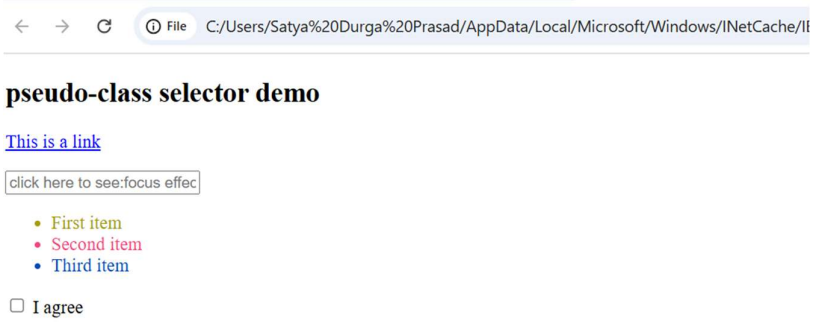
These pseudo-classes enhance user experience by providing visual feedback and structural styling based on element states and positions.

Program:

```
<!DOCTYPE html>
<html>
  <head>
        <style>
            /*Links*/
                a:link{
                  color:blue;
                }
                a:visisted{
                  color:purple;
                }
                a:hover{
                  color:red;
                }
                a:active{
                  color:green;
                }
```

```
        /*Input focus*/
        input:focus{
          border:2px solid orange;
          background-color:#cff2cb;
        }
                /*list styling*/
                ul li:first-child{
                   color: #a69c0d;
                }
                ul li:last-child{
                   color: #064ab8;
                }
                ul li:nth-child(2){
                   color: #f24680;
                }
        /* Check Boxes */
        input:checked + label{
             color:#118207;
             font-weight:bold;
        }
        </style>
  </head>
        <body>
                <h2>pseudo-class selector demo</h2>

                <p><a href="#">This is a link</a></p>

                <p><input type="text" placeholder="click here to see:focus effect"></p>

                <ul>
                  <li>First item</li>
                  <li>Second item</li>
                  <li>Third item</li>
                </ul>

                <p>
                  <input type="checkbox" id="check1">
                  <label for="check1">I agree</label>
                </p>
        </body>
</html>
```

Output:



Result:

4.a.iv) Pseudo-element selector

Aim

To demonstrate the use of CSS pseudo-element selectors that style specific parts of elements or insert content dynamically.

Description

This program illustrates various CSS pseudo-element selectors applied to HTML elements:

- p::first-letter styles the first letter of a paragraph with a large, bold, dark red font, enhancing visual emphasis.
- p::first-line styles the first line of a paragraph in italic and purple, improving readability and design.
- h2::before inserts a decorative star (⭐) before the heading with a greenish hue.
- h2::after appends a star (⭐) after the heading in cyan, creating symmetry and flair.
- ::selection customizes the appearance of selected text with a yellow background and black font, improving user interaction feedback.

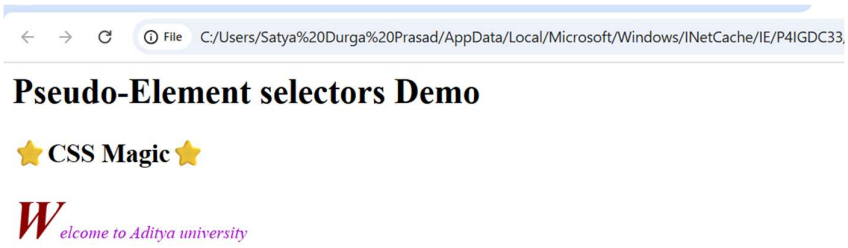
These pseudo-elements allow fine-grained control over content presentation and user experience without altering the HTML structure.

Program:

```
<DOCTYPE! html>
<html>
  <head>
    <style>
      /* ::first-letter-style the first letter of a paragraph */
      p::first-letter{
          font-size:50px;
          font-weight:bold;
          color:darkred;
        }
      /* ::first-line-style the first line of a paragraph */
      p::first-line{
          font-style: italic;
          color:#b004d6;
        }
      /* ::before-insert content before an element */
      h2::before{
        content:" ⭐ ";
        color:#95c408;
       }
      /* ::after-insert content after an element */
      h2::after{
```

```
        content:"⭐";
        color:#08c4c1;
      }
    /* ::selection-style selected text */
    ::selection{
      background-color: yellow;
      color: black;
    }
  </style>
 </head>
<body>
  <h1>Pseudo-Element selectors Demo</h1>
  <h2>CSS Magic</h2>
  <p>Welcome to Aditya university</p>
</body>
</html>
```

Output:



Result:

4.a.v) Attribute selector

Aim:

To demonstrate the use of CSS attribute selectors to style HTML elements based on specific attributes and attribute values.

Description:

This HTML document illustrates three types of CSS attribute selectors:

1. Type Attribute Selector

- Syntax: input[type="text"] { border: 2px solid blue; }
- Effect: Applies a blue border to elements with type="text".
- Styled:
- Not styled:

2. Target Attribute Selector

- Syntax: a[target="_blank"] { color: red; }
- Effect: Styles anchor () tags that open in a new tab with red text.
- Styled:
- Not styled:

3. Presence Attribute Selector

- Syntax: p[title] { background-color: yellow; }
- Effect: Highlights

elements that have a title attribute with a yellow background.

- Styled:
- Not styled:

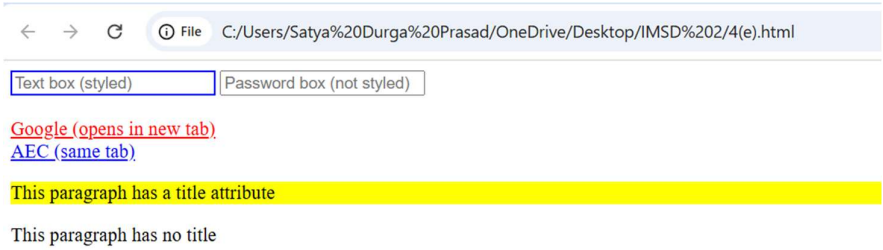
without title

Program:

```
<!DOCTYPE html>
<html>
<head>
   <title>Attribute Selectors</title>
   <style>
      /* Selects input with type="text" */
      input[type="text"] {
         border: 2px solid blue;
      }
```

```
        /* Selects links that open in new tab */
        a[target="_blank"] {
            color: red;
        }

        /* Selects elements having title attribute */
        p[title] {
            background-color: yellow;
        }
    </style>
</head>
<body>
    <input type="text" placeholder="Text box (styled)">
    <input type="password" placeholder="Password box (not styled)">

    <br><br>
    <a href="https://www.google.com" target="_blank">Google (opens in new tab)</a>
    <br>
    <a href="https://www.aec.edu.in" target="_self">AEC (same tab)</a>

    <p title="info">This paragraph has a title attribute</p>
    <p>This paragraph has no title</p>
</body>
</html>
```

Output:



Result:

Week-5

. CSS with Color, Background, Font, Text and CSS Box Model

5.a) Write a program to demonstrate the various ways you can reference a color in CSS.

Aim:

To demonstrate different methods of specifying colors in CSS, including named colors, hexadecimal values, RGB, RGBA, HSL, and HSLA formats**.**

Description:

This program demonstrates how CSS supports multiple color formats, each with its own syntax and use case:

- **Named Colors:** Predefined color names like tomato, blue, or gold for quick styling.
- **Hexadecimal (#RRGGBB):** Compact and widely used in web design, representing red, green, and blue values**.**
- **RGB (rgb(r, g, b)):** Explicit control over red, green, and blue channels using numeric values.
- **RGBA (rgba(r, g, b, a)):** Adds an alpha channel for transparency, useful for overlays and effects.
- **HSL (hsl(hue, saturation, lightness)):** More intuitive for designers, allowing control over color tone and brightness.
- **HSLA (hsla(h, s, l, a)):** Adds transparency to HSL, combining intuitive color control with visual layering.

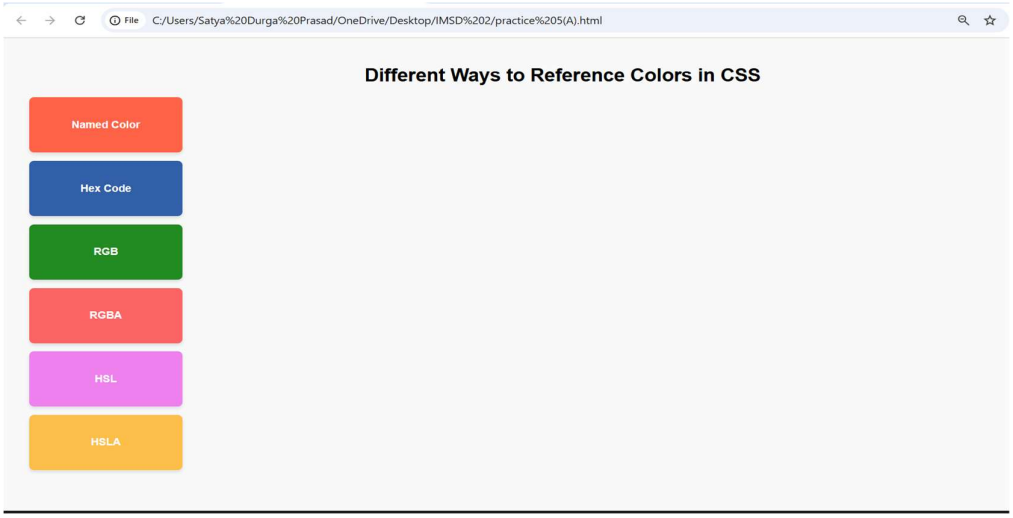
Program:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>CSS Color Reference Methods</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      padding: 20px;
      background-color: #f9f9f9;
    }

    h1 {
      text-align: center;
    }

    .color-box {
```

```
      width: 250px;
      height: 100px;
      color: white;
      display: flex;
      align-items: center;
      justify-content: center;
      margin: 15px;
      font-size: 18px;
      font-weight: bold;
      border-radius: 8px;
      box-shadow: 0 4px 6px rgba(0,0,0,0.1);
    }

    /* 1. Named color */
    .named-color {
      background-color: tomato; /* CSS Named Color */
    }

    /* 2. Hexadecimal notation */
    .hex-color {
      background-color: #315fa8; /* Dodger Blue */
    }

    /* 3. RGB notation */
    .rgb-color {
      background-color: rgb(34, 139, 34); /* Forest Green */
    }

    /* 4. RGBA notation (with transparency) */
    .rgba-color {
      background-color: rgba(255, 0, 0, 0.6); /* Red with transparency */
    }

    /* 5. HSL notation */
    .hsl-color {
      background-color: hsl(300, 76%, 72%); /* Light purple */
    }

    /* 6. HSLA notation (with transparency) */
    .hsla-color {
      background-color: hsla(39, 100%, 50%, 0.7); /* Orange with transparency */
    }
  </style>
</head>
<body>
```

```
    <h1>Different Ways to Reference Colors in CSS</h1>

    <div class="color-box named-color">Named Color</div>
    <div class="color-box hex-color">Hex Code</div>
    <div class="color-box rgb-color">RGB</div>
    <div class="color-box rgba-color">RGBA</div>
    <div class="color-box hsl-color">HSL</div>
    <div class="color-box hsla-color">HSLA</div>
</body>
</html>
```

Output:



Result:

5.b) Write a CSS rule that places a background image halfway down the page, tilting it horizontally. The image should remain in place when the user scrolls up or down.

Aim:

To create a visually striking background image that is fixed halfway down the viewport, tilted horizontally, and remains stationary as the user scrolls through the page.

Description:

This program showcases a visually engaging technique where a background image is:

- Fixed in the viewport using position: fixed
- Tilted using transform: rotateZ(-12deg)
- Centered both horizontally and vertically with translate(-50%, -50%)
- Styled with rounded corners and a subtle shadow for depth
- Non-interactive, thanks to pointer-events: none, so it doesn't interfere with page content

The rest of the page is scrollable (min-height: 300vh), allowing users to observe how the tilted image remains anchored in place while content flows around it.

Program:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<title>Fixed tilted background (horizontal rotate)</title>
<style>
 /* Container content so you can scroll the page */
 body {
  min-height: 300vh; /* make page scrollable to test fixed behaviour */
  margin: 0;
  font-family: system-ui, Arial, sans-serif;
 }
 /* Fixed background image element */
 .tilted-bg {
  position: fixed;
  left: 50%;          /* center horizontally */
  top: 50vh;          /* halfway down the viewport */
  transform: translate(-50%, -50%) rotateZ(-12deg) translateZ(0);
                  /* translate to center, then rotate; translateZ(0) triggers GPU */
  width: 40vw;        /* size the image element (adjust as needed) */
  height: 40vh;
```

```
   background-image:
url('https://res.cloudinary.com/dvsumlmle/image/upload/v1756891540/dp_aditya_dbunou.jpg
');
   background-size: cover;
   background-position: center;
   background-repeat: no-repeat;
   pointer-events: none; /* so it won't block clicks */
   border-radius: 12px;
   box-shadow: 0 10px 30px rgba(0,0,0,0.25);
   z-index: 99
99;       /* keep it on top (adjust if necessary) */
   capacity: 0.95;
   will-change: transform;
  }
  /* sample page content */
  main {
   padding: 2rem;
  }
</style>
</head>
<body>
  <div class="tilted-bg" aria-hidden="true"></div>
  <main>
   <h1>Page Content</h1>
   <p>Scroll to see the tilted background remain fixed halfway down the viewport.</p>
   <p>Adjust <code>width</code>, <code>height</code>, and rotation angle in the CSS as
needed.</p>
   <!-- Add content to scroll -->
   <p style="margin-top:140vh">This line is roughly below the fold to show scrolling.</p>
  </main>
</body>
</html>
```

Output:

This line is roughly below the fold to show scrolling.

Result:

5.c) Write a program using the following terms related to CSS font and text: i. font-size ii. font-weight iii. font-style iv. text-decoration v. text-transformation vi. text-alignment

Aim:

To create a simple web page that demonstrates the use of key CSS font and text properties—including font-size, font-weight, font-style, text-decoration, text-transform, and text-align—by applying them to different text elements.

Description:

This HTML and CSS program showcases six key text-related styling techniques:

- font-size: Adjusts the size of the text for emphasis or readability.
- font-weight: Controls the thickness of the font, such as making it bold.
- font-style: Applies styles like italic to differentiate or highlight text.
- text-decoration: Adds visual effects like line-through, underline, or overline.
- text-transform: Alters the case of text (e.g., uppercase for emphasis).
- text-align: Aligns text horizontally within its container (e.g., center alignment).

Each

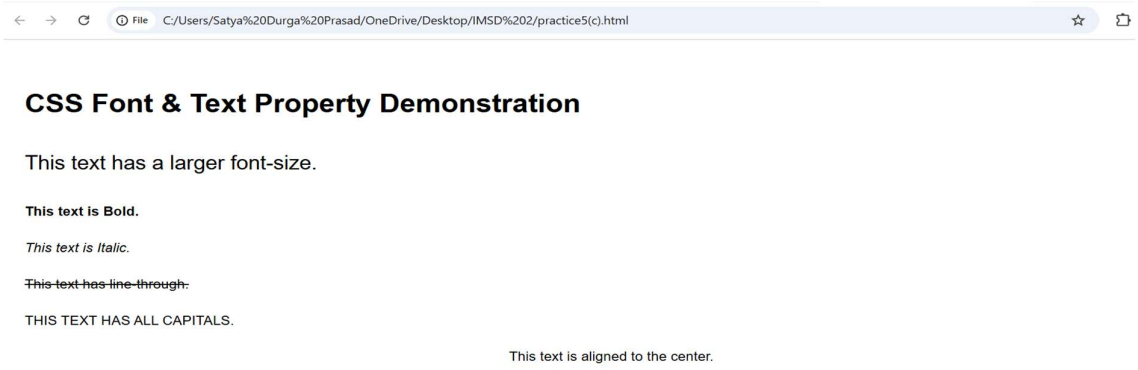
element is styled with a different property to visually demonstrate its effect. This is useful for understanding how CSS can enhance typography and layout in web design.

Program:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>CSS Font & Text Properties</title>
  <style>
    body {
      font-family: Arial, "Bree Serif", serif;
      padding: 20px;
      line-height: 1.8;
    }
    .font-size {
      font-size: 24px;
    }
    .font-weight {
      font-weight: bold; /* fixed: it was lighter, but text says "Bold" */
    }
    .text-decoration {
      text-decoration: line-through;
```

```
        }
        .text-align {
            text-align: center;
        }
        .font-style {
            font-style: italic;
        }
        .text-transform {
            text-transform: uppercase; /* fixed: text says "all capitals" */
        }
    </style>
</head>
<body>
    <h1>CSS Font & Text Property Demonstration</h1>

    <p class="font-size">This text has a larger font-size.</p>
    <p class="font-weight">This text is Bold.</p>
    <p class="font-style">This text is Italic.</p>
    <p class="text-decoration">This text has line-through.</p>
    <p class="text-transform">This text has all capitals.</p>
    <p class="text-align">This text is aligned to the center.</p>
</body>
</html>
```

Output:



Result:

5.d) Write a program, to explain the importance of CSS Box model using i. Content ii. Border iii. Margin iv. Padding

Aim

To demonstrate the CSS box model by visually representing its key components—content, padding, border, and margin—using a styled

element.

Description

The CSS box model is a fundamental concept in web design that defines how elements are structured and spaced. This program illustrates each part:

- **Content:** The actual text or media inside the box (background-color: lightblue).
- **Padding**: The space between the content and the border (padding: 20px).
- **Border:** The visible outline around the padding (border: 5px solid darkgreen).
- **Margin:** The space outside the border that separates the box from other elements (margin: auto centers it horizontally).

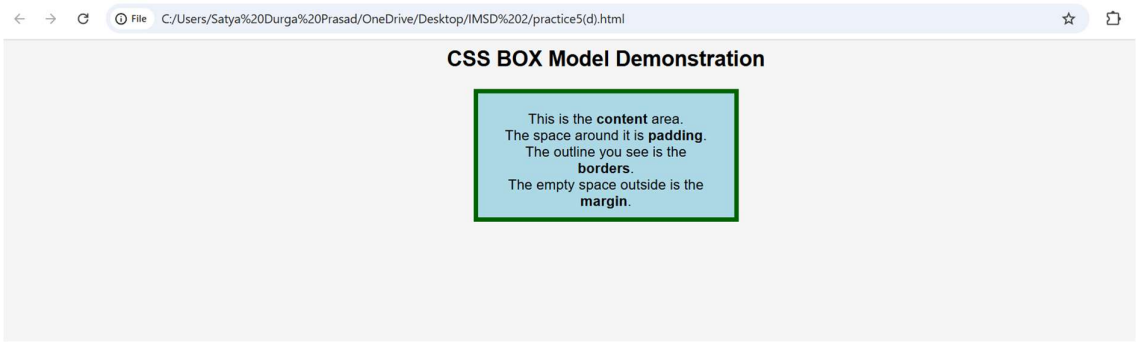
The layout uses a fixed width and height for clarity, and the explanatory text inside the box reinforces each concept. The background and font styling enhance readability and visual appeal.

Program:

```
<DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>CSS box model Example</title>
    <style>
       .box {
          width: 250px;
          height: 100px;
          background-color: lightblue;    /*i.content*/
          padding: 20px;                  /*iv.padding*/
          border: 5px solid darkgreen;     /*ii.Border*/
          margin: auto;                   /*iii.Margin*/
          font-size: 16px;
          text-align: center;
       }
      body {
```

```
        background-color: #f5f5f5;
        font-family:Arial,sans-serif;
      }
    </style>
</head>
<body>
    <h2 style="text-align:center;">CSS BOX Model Demonstration</h2>
    <div class="box">
        This is the <b>content</b> area.<br>
        The space around it is <b>padding</b>.<br>
        The outline you see is the <b>borders</b>.<br>
        The empty space outside is the <b>margin</b>.
    </div>
</body>
</html>
```

Output:



Result:

Week 6

6.a)Write a program to embed internal and external JavaScript in a web page.

Aim:

To demonstrate embedding internal and external JavaScript in a single HTML web page.

Description:

JavaScript can be added to a web page in two main ways:

Internal JavaScript: Written directly inside the HTML file using the <script> tag

External JavaScript: Stored in a separate .js file and linked to the HTML using the src attribute in the <script> tag

This program shows both methods: an internal script that displays a greeting and an external script that changes the background color.

Program:

Internal javascript:

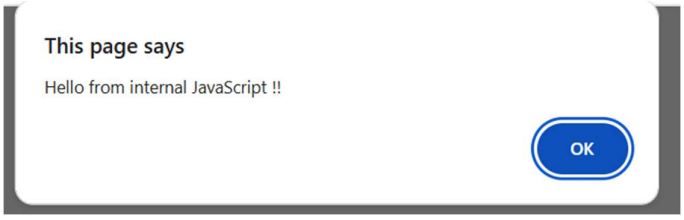
```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>JavaScript Embedding Example</title>
  <!--External Java script--!>
  <script src="6(A).js"></script>
</head>
<body>

    <h1>Welcome to JavaScript Demo</h1>
    <button onclick="internalFunction()">Run Internal JS</button>
    <button onclick="externalFunction()">Run External JS</button>

   <!--Internal JavaScript-->

   <script>
     function internalFunction(){
          alert("Hello from internal JavaScript !!");
     }
   </script>
     <script>

</body>
</html>
```

External Javascript:

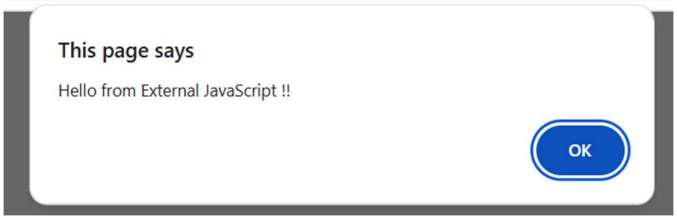
```
function externalFunction(){
        alert("Hello from External JavaScript !!");
    }
```

Output:

Internal javascript:



External javascript:



Result:

6.b)Write a program to explain the different ways for displaying output.

Aim:

To demonstrate various JavaScript output methods including alert boxes, console logging, document writing, and DOM manipulation**.**

Description

JavaScript provides multiple ways to display output to users or developers:

- **Alert Box (alert):** Displays a popup message to the user.
- **Console Log (console.log):** Sends output to the browser's developer console, useful for debugging.
- **Document Write (document.write):** Directly writes content to the HTML document (not recommended for dynamic pages).
- **DOM Manipulation (innerText, innerHTML):** Updates specific elements on the page dynamically.

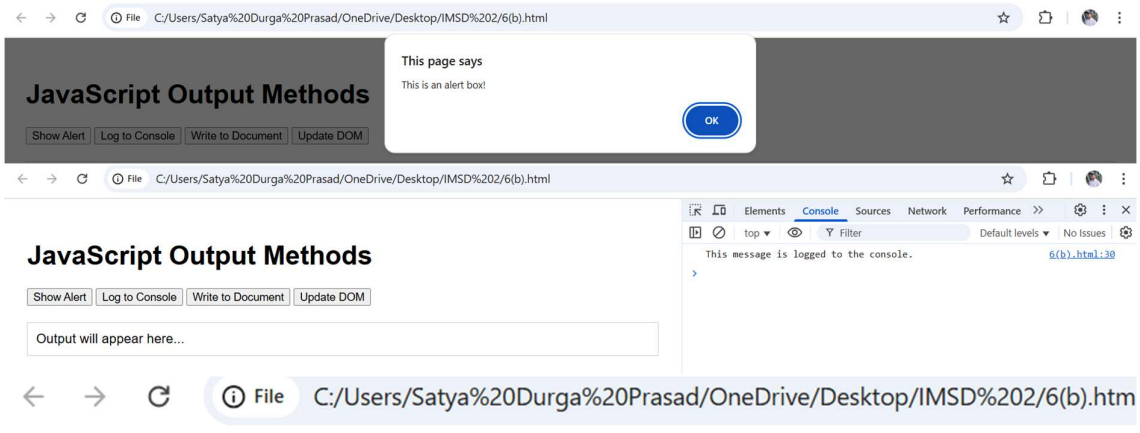
This program includes four buttons, each triggering one of these output methods. It helps visualize how JavaScript interacts with the browser and the document structure.

Program:

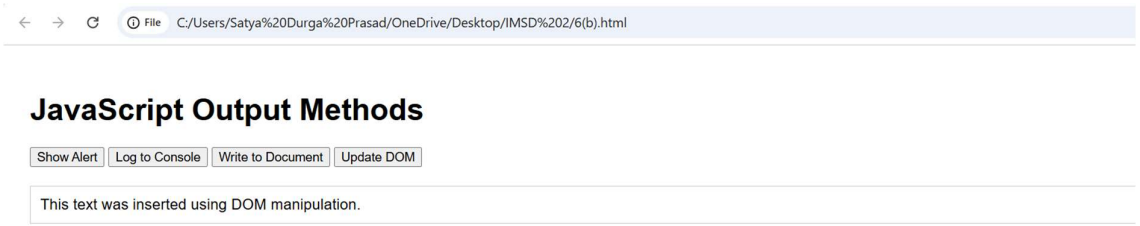
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>JavaScript Output Methods</title>
  <style>
    body { font-family: Arial, sans-serif; padding: 20px; }
    #outputArea { margin-top: 20px; padding: 10px; border: 1px solid #ccc; }
  </style>
</head>
<body>
  <h1>JavaScript Output Methods</h1>
  <button onclick="showAlert()">Show Alert</button>
  <button onclick="logToConsole()">Log to Console</button>
  <button onclick="writeToDocument()">Write to Document</button>
  <button onclick="updateDOM()">Update DOM</button>
  <div id="outputArea">Output will appear here...</div>
  <script>
    // 1. Alert box
    function showAlert
() {
```

```
    alert("This is an alert box!");
  }
  // 2. Console log
  function logToConsole() {
    console.log("This message is logged to the console.");
  }
  // 3. Document write
  function writeToDocument() {
    document.write("This text is written directly to the document.");
  }
  // 4. DOM manipulation
  function updateDOM() {
    document.getElementById("outputArea").innerText = "This text was inserted using
DOM manipulation.";
  }
 </script>
</body>
</html>
```

Output:



This text is written directly to the document.


File
C:/Users/Satya%20Durga%20Prasad/OneDrive/Desktop/IMSD%202/6(b).html
JavaScript Output Methods
Show Alert
Log to Console
Write to Document
Update DOM
This text was inserted using DOM manipulation.

Result:

6.c) Write a program to explain the different ways for taking input.

Aim

To demonstrate various methods of taking user input in JavaScript using HTML elements and event handling.

Description

This program showcases four distinct input techniques in JavaScript:

1. **Text Input Field**
    - Users enter their name in a text box.
    - On clicking the "Submit Name" button, the value is retrieved using getElementById().value and displayed.
2. **Prompt Box**
    - Clicking the "Enter Age (prompt)" button triggers a JavaScript prompt() dialog.
    - The entered age is captured and shown in the output area.
3. **Form Input**
    - Users enter their email in a form field.
    - On form submission, the default reload is prevented using event.preventDefault().
    - The email is extracted and displayed.
4. **Dropdown Selection**

- A dropdown menu allows users to select a country.
- On clicking "Submit Country", the selected value is retrieved and shown.

All inputs are dynamically displayed in the

with ID output, providing immediate feedback to the user.

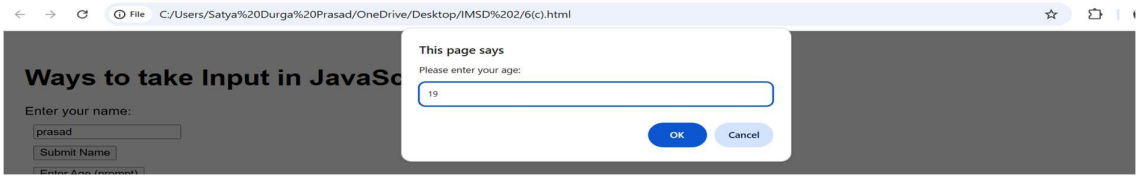
Program:

```
<!DOCTYPE html>
<html>
  <head>
    <title>JavaScript Input methods</title>
      <style>
        body{
            font-family:Arial;
            padding:20px;
        }
```

```
            input,button,select{
                margin:10px;
                display:block;
            }
            #output{
                margin-top:20px;
                font-weight:bold;
            }
        </style>
    </head>
<body>
    <h1>Ways to take Input in JavaScript</h1>
    <!-- 1.Text Input Field-->
    <label>Enter your name:</label>
    <input type="text" id="nameInput">
    <button onclick="getInputFromField()">Submit Name</button>

    <!--2.Prompt Box-->
    <button onclick="getInputFromPrompt()">Enter Age (prompt)</button>

    <!--3.Form Input-->
    <form onsubmit="getInputFromForm(event)">
        <label>Enter your email:</label>
        <input type="email" id="emailInput" required>
        <button type="submit">Submit Email</button>
    </form>

    <!--4.Dropdown Selection-->
    <label>Select your country:</label>
    <select id="countrySelect">
        <option value="India">India</option>
        <option value="USA">USA</option>
        <option value="Germany">Germany</option>
    </select>
    <button onclick="getInputFromDropdown()">Submit Country</button>

    <div id="output">Your input will appear here.</div>

        <script>
            //1. Input from text field
            function getInputFromField(){
                const name=document.getElementById("nameInput").value;
                document.getElementById("output").innerText="Name: "+name;
            }

            //2. Input from prompt box
```

```
        function getInputFromPrompt(){
                const age=prompt("Please enter your age:");
                document.getElementById("output").innerText="Age: "+age;
            }

        //3. Input from Form
        function getInputFromForm(event){
                event.preventDefault(); // Prevent form reload
                const email=document.getElementById("emailInput").value;
                document.getElementById("output").innerText="Email: "+email;
            }

        //4. Input from Dropdown
        function getInputFromDropdown(){
                const country=document.getElementById("countrySelect").value;
                document.getElementById("output").innerText="Country: "+country;
            }
    </script>
</body>
</html>
```

Output:



Ways to take Input in JavaScript

Enter your name:

prasad

Submit Name

Enter Age (prompt)

Enter your email:

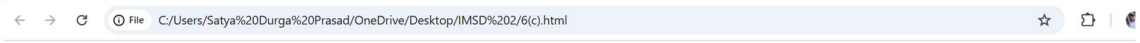
Submit Email

Select your country:

India

Submit Country

Age: 19



Ways to take Input in JavaScript

Enter your name:

prasad

Submit Name

Enter Age (prompt)

Enter your email:

pinisetty@gmail.com

Submit Email

Select your country:

India

Submit Country

Email: pinisetty@gmail.com

Ways to take Input in JavaScript

Enter your name:

prasad

Submit Name

Enter Age (prompt)

Enter your email:

pinisetty@gmail.com

Submit Email

Select your country:

India

Submit Country

Country: India

Result:

6.d) Create a webpage which uses prompt dialogue box to ask a voter for his name and age. Display the information in table format along with either the voter can vote or not.

Aim

To build a JavaScript-based web application that checks voter eligibility by validating user input and displaying the result in a formatted table**.**

Description

This program prompts the user for their name and age, validates the inputs, and displays the eligibility status in a table.

- Features:
  - **Input via prompt():**
    - Name: must contain only letters and spaces (/^[A-Za-z\s]+$/)
    - Age: must be a valid number
  - **Validation Logic:**
    - Rejects empty or null inputs
    - Rejects names with numbers or special characters
    - Rejects non-numeric age entries
  - **Eligibility Check:**
    - Age ≥ 18 → "Eligible to vote"
    - Age < 18 → "Not Eligible to vote"
  - **Output:**
  - A styled HTML table displays:
  - Name
  - Age
  - Eligibility

Visuals:

- Table uses blue header styling with white text.
- Font is clean and readable (Arial).
- Output appears below the button upon successful validation

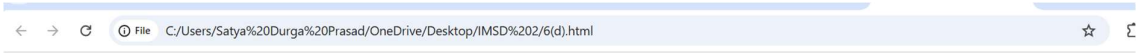
Program:

<!DOCTYPE html>

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Voter Eligibility Check</title>
    <style>
      body {
        font-family: Arial, sans-serif;
        padding: 20px;
      }
      table {
        border-collapse: collapse;
        width: 50%;
        margin-top: 20px;
      }
      th, td {
        border: 1px solid #333;
        padding: 10px;
        text-align: center;
      }
      th {
         background-color: blue;
         color: white;
      }
    </style>
</head>
<body>

    <h1>Voter Eligibility Checker</h1>
    <button onclick="checkvoter()">Check Eligibility</button>

    <div id="result"></div>

    <script>
       function checkvoter() {
          const name = prompt("Enter your name:");
          const ageInput = prompt("Enter your age:");

          // Regular expression: allows only letters and spaces
          const nameRegex = /^[A-Za-z\s]+$/;

          if (name === null || ageInput === null ||
             name.trim() === "" ||
             !nameRegex.test(name) ||  // <-- prevents numbers in name
             isNaN(ageInput)) {
             alert("Invalid input. Please try again.");
             return;
```

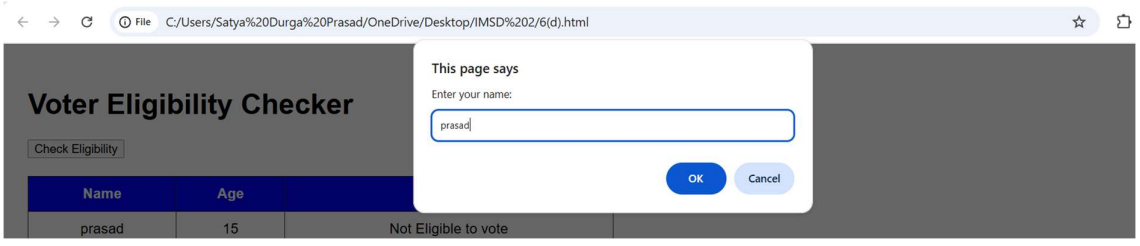
```
        }

        const ageNum = parseInt(ageInput);
        const eligibility = ageNum >= 18 ? "Eligible to vote" : "Not Eligible to vote";

        const tableHTML = `
          <table>
            <tr>
              <th>Name</th>
              <th>Age</th>
              <th>Eligibility</th>
            </tr>
            <tr>
              <td>${name}</td>
              <td>${ageNum}</td>
              <td>${eligibility}</td>
            </tr>
          </table>
        `;
        document.getElementById("result").innerHTML = tableHTML;
      }
    </script>
</body>
</html>
```

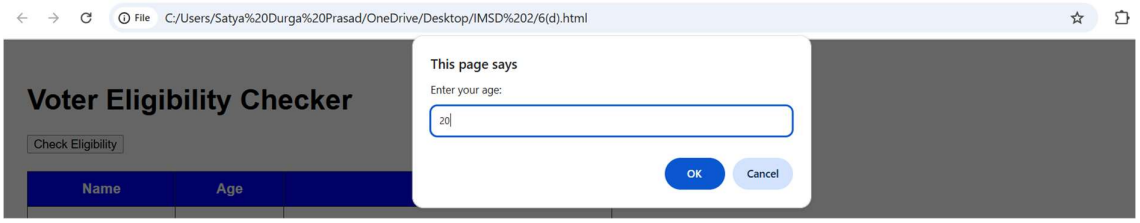
Output:

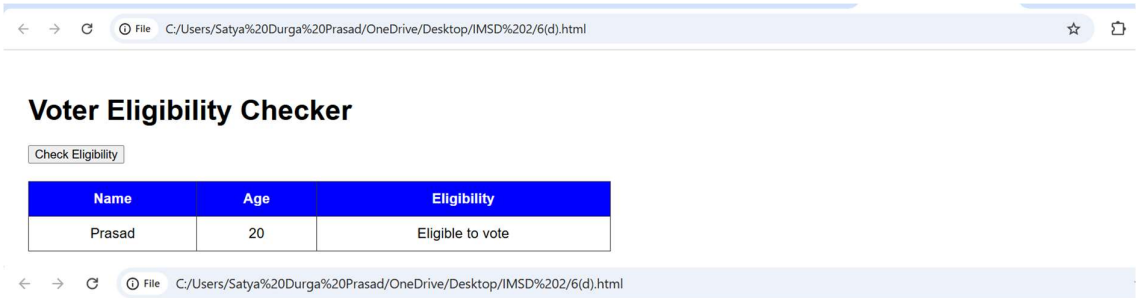


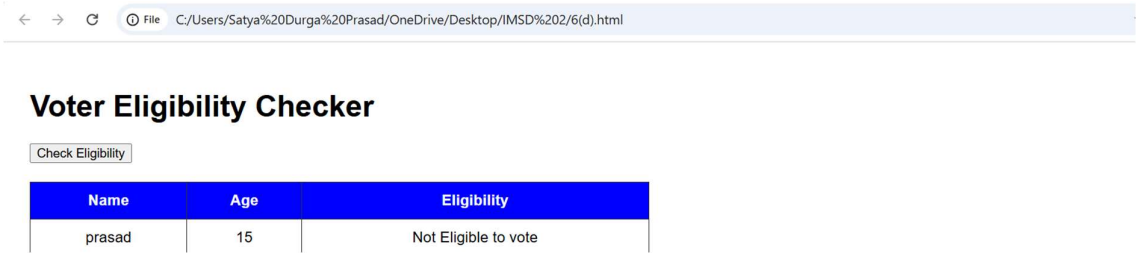
Voter Eligibility Checker

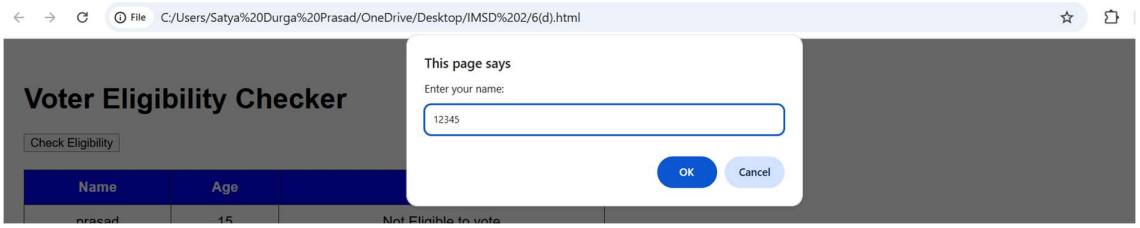
Check Eligibility

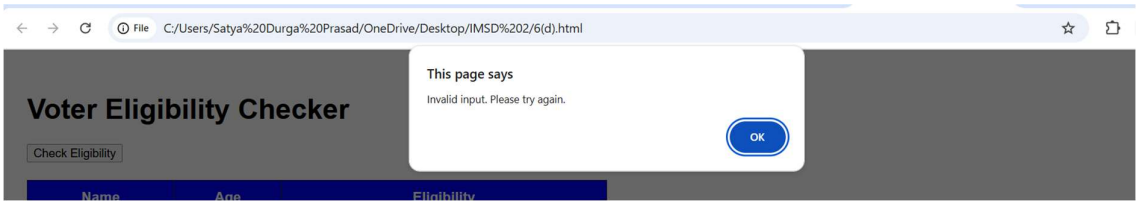





File C:/Users/Satya%20Durga%20Prasad/OneDrive/Desktop/IMSD%202/6(d).html
Voter Eligibility Checker
Check Eligibility
Name
Age
Eligibility
Prasad
20
Eligible to vote


File C:/Users/Satya%20Durga%20Prasad/OneDrive/Desktop/IMSD%202/6(d).html
Voter Eligibility Checker
Check Eligibility
Name
Age
Eligibility
prasad
15
Not Eligible to vote


File C:/Users/Satya%20Durga%20Prasad/OneDrive/Desktop/IMSD%202/6(d).html
Voter Eligibility Checker
Check Eligibility
Name
Age
This page says
Enter your name:
12345
OK
Cancel


File C:/Users/Satya%20Durga%20Prasad/OneDrive/Desktop/IMSD%202/6(d).html
Voter Eligibility Checker
Check Eligibility
This page says
Invalid input. Please try again.
OK

Result:

Week – 7

7.a) Write a program using document object properties and methods.

Aim:

To demonstrate the use of the Document Object Model (DOM) in JavaScript by accessing document properties and manipulating HTML elements dynamically.

Description

This webpage showcases how JavaScript interacts with the DOM:

- **HTML Structure:**
    - A heading (

) with the ID heading.

    - A paragraph explaining the demo.
    - A button labeled "Show document Info" that triggers a JavaScript function.
    - A

with ID Output to display document details.

- **JavaScript Functionality (showInfo()):**
- Accesses document-level properties:
    - document.title: Gets the title of the webpage.
    - document.URL: Retrieves the current page URL.
    - document.lastModified: Shows the last modified timestamp of the document.
- Dynamically changes the heading color to blue using getElementById.
- Displays the retrieved document information inside the Output

using innerHTML.

Program:

```
<html>
<head>
    <title>Document Object Example</title>
</head>
<body>
    <h1 id="heading">Hello World</h1>
    <p>This is a demo of document object.</p>
    <button onclick="showInfo()">Show document Info</button>
    <div id="Output"></div>
    <script>
        function showInfo() {
```

```
        //Using document properties
        let title = document.title;
        let url = document.URL;
        let lastModified = document.lastModified;

        //using document methods
        document.getElementById("heading").style.color = "blue";
      //change heading color
        document.getElementById("Output").innerHTML =
          "Title:" + title + "<br>" +
          "URL:" + url + "<br>" +
          "Last Modified:" + lastModified;
    }
   </script>
</body>
</html>
```

Output:

Result:

7.b) Write a program using window object properties and methods.

Aim:

To demonstrate the use of the Window Object in JavaScript for accessing browser window properties and controlling new browser windows.

Description

This webpage showcases how JavaScript interacts with the browser's window object:

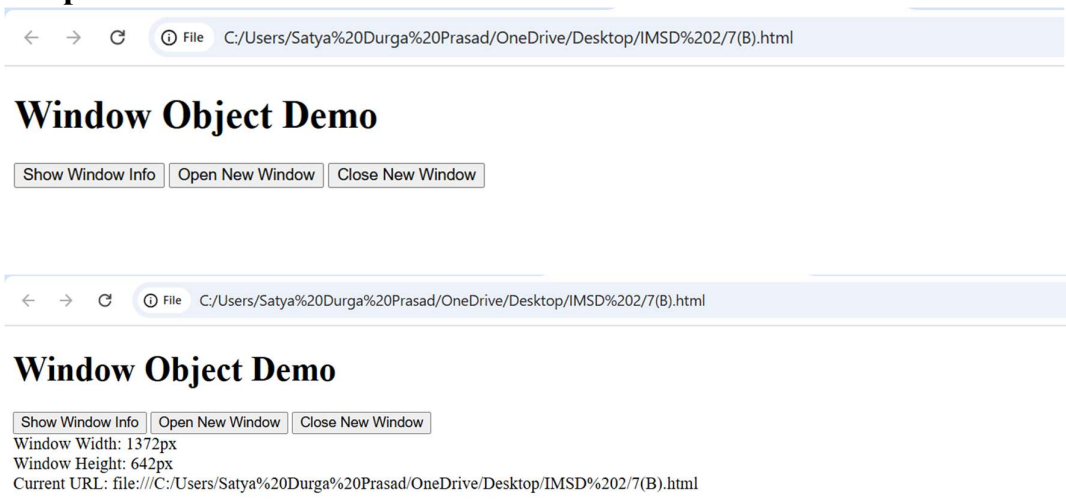
- **HTML Elements:**
    - A heading (<h1>) titled *Window Object Demo*.
    - Three buttons:
        - **Show Window Info**: Displays current window dimensions and URL.
        - **Open New Window**: Opens a new browser window with Amazon's website.
        - **Close New Window**: Closes the newly opened window.
- **JavaScript Functionality:**
- **showWindowInfo():**
    - Uses window.innerWidth and window.innerHeight to get the current window size.
    - Uses window.location.href to get the current page URL.
    - Displays this info inside the Output
- **openNewWindow():**
    - Uses window.open() to launch a new window with specified dimensions (400×300) and loads Amazon.
    - Stores the reference in newWin for later control.
- **closeNewWindow():**
- Checks if newWin exists and closes it using newWin.close().

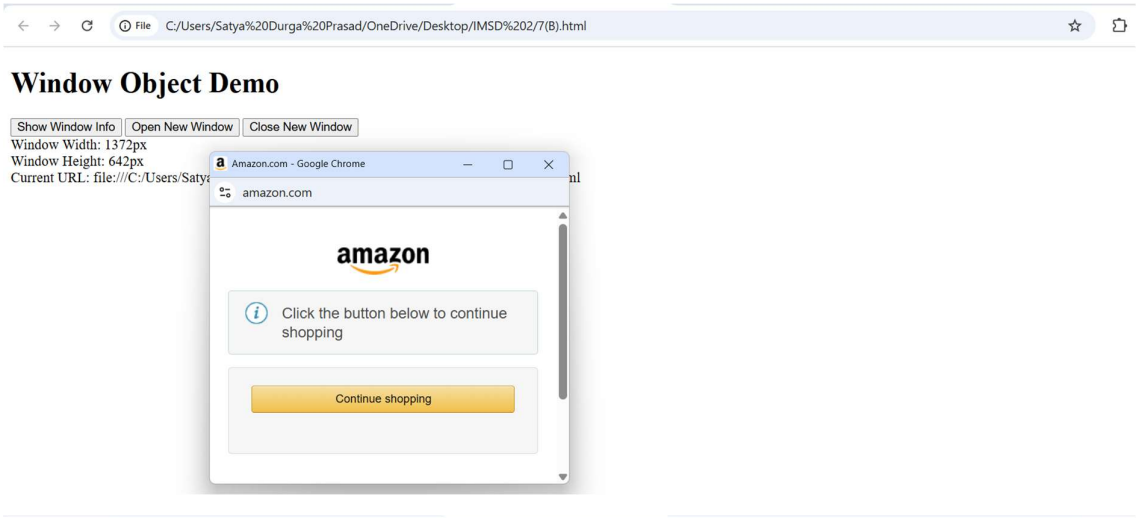
Program:

```
<!DOCTYPE html>
<html>
<head>
   <title>Window Object Example</title>
</head>
<body>
```

```
    <h1>Window Object Demo</h1>
    <button onclick="showWindowInfo()">Show Window Info</button>
    <button onclick="openNewWindow()">Open New Window</button>
    <button onclick="closeNewWindow()">Close New Window</button>
    <div id="Output"></div>
    <script>
        let newWin;
        function showWindowInfo() {
            let width = window.innerWidth;
            let height = window.innerHeight;
            let locationHref = window.location.href;
            document.getElementById("Output").innerHTML = "Window Width: " + width +
"px<br>" + "Window Height: " + height + "px<br>" + "Current URL: " + locationHref;
        }
        function openNewWindow() {
            newWin = window.open("https://www.amazon.com", "Example",
"width=400,height=300");
        }
        function closeNewWindow() {
            if (newWin) {
                newWin.close();
            }
        }
    </script>
</body>
</html>
```

Output:





Window Object Demo

Show Window Info | Open New Window | Close New Window

Window Width: 1372px
Window Height: 642px
Current URL: file:///C:/Users/Satya%20Durga%20Prasad/OneDrive/Desktop/IMSD%202/7(B).html

Result:

7.c) Write a program using array object properties and methods.

Aim:

To demonstrate various array methods and properties in JavaScript. It showcases how to interact with arrays by manipulating elements and using built-in methods such as push(), unshift(), pop(), shift(), sort(), and join().

Description:

1. **Array Declaration:**
    - An array fruits is initialized with three elements: "Apple", "Banana", and "Mango".
2. **Length Property:**
    - The length property is used to get the number of elements in the array initially, which is displayed on the page.
3. **Array Methods:**
    - **push("Orange"):** Adds "Orange" to the end of the array.
    - **unshift("Grapes"):** Adds "Grapes" to the beginning of the array.
    - **pop():** Removes and returns the last element of the array ("Orange").
    - **shift():** Removes and returns the first element of the array ("Grapes").
    - **sort():** Sorts the array alphabetically.
    - **join():** Joins the elements of the array into a string, separated by commas

Program:

```
<!DOCTYPE html>
<html>
<head>
   <title>Array Object Properties and Methods</title>
</head>
<body>
   <h2>Array Object Example</h2>
   <p id="Output"></p>
   <script>
      let fruits = ["Apple", "Banana", "Mango"];
      let length = fruits.length;
      fruits.push("Orange");
      fruits.unshift("Grapes");
      let last = fruits.pop();
      let first = fruits.shift();
      let sorted = fruits.sort();
      let joined = fruits.join();
```

```
        document.getElementById("Output").innerHTML = "<b>Original length:</b> " +
length + "<br>" +
            "<b>After Push & Unshift:</b> Grapes, Apple, Banana, Mango, Orange<br>" +
            "<b>Removed Last:</b> " + last + "<br>" +
            "<b>Removed First:</b> " + first + "<br>" +
            "<b>Sorted Array:</b> " + sorted + "<br>" +
            "<b>Joined as String:</b> " + joined;
    </script>
</body>
</html>
```

Output:

Array Object Example

Original length: 3
After Push & Unshift: Grapes, Apple, Banana, Mango, Orange
Removed Last: Orange
Removed First: Grapes
Sorted Array: Apple,Banana,Mango
Joined as String: Apple,Banana,Mango

Result:

7.d) Write a program using math object properties and methods.

Aim:

To demonstrate the use of the JavaScript Math object, including its properties and commonly used methods for mathematical operations.

Description:

This HTML document uses JavaScript to illustrate various properties and methods of the Math object. It displays:

- Math properties such as PI, E, and SQRT2, which represent important mathematical constants.
- Math methods like:
    - **Math.sqrt()** to calculate square roots,
    - **Math.pow()** for exponentiation,
    - **Math.abs()** for absolute value,
    - **Math.round(),** Math.ceil(), and Math.floor() for different types of rounding,
    - **Math.random()** to generate a random number between 0 and 1,
    - **Math.max()** and **Math.min()** to find the maximum and minimum values among a set of numbers.

The output is dynamically displayed inside a paragraph element with the ID Output using innerHTML.

Program:

```
<!DOCTYPE html>
<html>
<head>
   <title>Math Object Properties and Methods</title>
</head>
<body>
   <h2>Math Object Example</h2>
   <p id="Output"></p>
   <script>
      let sqrtResult = Math.sqrt(26);
      let powerResult = Math.pow(2, 5);
      let absResult = Math.abs(-25);
      let roundResult = Math.round(4.7);
      let ceilResult = Math.ceil(4.2);
      let floorResult = Math.floor(4.8);
      let randomResult = Math.random();
      let maxResult = Math.max(3, 9, 2, 15, 7);
      let minResult = Math.min(3, 9, 2, 15, 7);
```

```
        document.getElementById("Output").innerHTML = "<b>Properties:</b>" +
            "PI: " + Math.PI + "<br>" +
            "E: " + Math.E + "<br>" +
            "SQRT2: " + Math.SQRT2 + "<br><br>" +
            "<b>Methods:</b><br>" +
            "Square Root of 26: " + sqrtResult + "<br>" +
            "2^5 (power): " + powerResult + "<br>" +
            "Absolute of -25: " + absResult + "<br>" +
            "Round(4.7): " + roundResult + "<br>" +
            "Ceil(4.2): " + ceilResult + "<br>" +
            "Floor(4.8): " + floorResult + "<br>" +
            "Random Number (0-1): " + randomResult + "<br>" +
            "Max(3,9,2,15,7): " + maxResult + "<br>" +
            "Min(3,9,2,15,7): " + minResult;
    </script>
</body>
</html>
```

Output:

Math Object Example

Properties:PI: 3.141592653589793
E: 2.718281828459045
SQRT2: 1.4142135623730951

Methods:
Square Root of 26: 5.0990195135927845
2^5 (power): 32
Absolute of -25: 25
Round(4.7): 5
Ceil(4.2): 5
Floor(4.8): 4
Random Number (0-1): 0.6312845290294098
Max(3,9,2,15,7): 15
Min(3,9,2,15,7): 2

Result:

7.e) Write a program using string object properties and methods.

Aim:

To demonstrate the use of the JavaScript String object, including its property and various string manipulation methods.

Description:

This HTML document showcases different features of the JavaScript String object using a sample text " Hello JavaScript World: ".

- **Property Used:**
  - .length: Returns the number of characters in the string.
- **Methods Demonstrated:**
  - **.toUpperCase():** Converts the string to uppercase.
  - **.toLowerCase():** Converts the string to lowercase.
  - .**trim():** Removes whitespace from both ends of the string.
  - **.slice(start, end):** Extracts a section of the string from index 6 to 16.
  - .**replace("World", "Programmers"):** Replaces a substring with another.
  - **.charAt(index):** Returns the character at a specific index (7).
  - .**indexOf("JavaScript"):** Finds the position of the word "JavaScript" in the string.
  - **.split(" "):** Splits the string into an array using space as a separator (after trimming).

The results are displayed on the web page inside a paragraph element with the ID Output using innerHTML.

Program:

```
<html>
<head>
</head>
<body>
   <h2>JavaScript String Object Example</h2>
   <p id="Output"></p>
   <script>
      // Create a string
      let text = " Hello JavaScript World: ";
      // Using String Properties
      let length = text.length; // property
      // Using String Methods
      let upper = text.toUpperCase();
```

```
        let lower = text.toLowerCase();
        let trimmed = text.trim();
        let sliced = text.slice(6, 16);
        let replaced = text.replace("World", "Programmers");
        let charAtPos = text.charAt(7);
        let index = text.indexOf("JavaScript");
        let splitted = text.trim().split(" ");
        // Display the results
        document.getElementById("Output").innerHTML =
            "<b>Original String:</b> " + text + "<br>" +
            "<b>Length:</b> " + length + "<br>" +
            "<b>Uppercase:</b> " + upper + "<br>" +
            "<b>Lowercase:</b> " + lower + "<br>" +
            "<b>Trimmed:</b> " + trimmed + "<br>" +
            "<b>Sliced (6, 16):</b> " + sliced + "<br>" +
            "<b>Replaced:</b> " + replaced + "<br>" +
            "<b>Character at 7:</b> " + charAtPos + "<br>" +
            "<b>Index of 'JavaScript':</b> " + index + "<br>" +
            "<b>Splitted:</b> " + splitted.join(", ");
    </script>
</body>
</html>
```

Output:

Result: