Alright 👍
Now I will **repeat Program–1 properly**, exactly the way you asked:

Program–1 Question (as per Lab Manual)

Develop the HTML page named as "Myfirstwebpage.html". Add the following tags with relevant content.

1. Set the title of the page as **"My First Web Page"**
2. Within the body use the following tags:
   a) Moving text
   b) Different heading tags (h1 to h6)
   c) Paragraph
   d) Horizontal line
   e) Line Break
   f) Block Quote
   g) Pre tag
   h) Different Logical Style (<b>, <u>, <sub>, <sup> etc.)

Program–1: Myfirstwebpage.html (Source Code)

```
<!DOCTYPE html>
<html>
<head>
  <title>My First Web Page</title>
</head>
<body>

  <!-- a) Moving text -->
  <marquee>Basic HTML Tags</marquee>

  <!-- b) Different heading tags -->
  <h1>Welcome to 5th Sem Students</h1>
  <h2>Welcome to 5th Sem Students</h2>
  <h3>Welcome to 5th Sem Students</h3>
  <h4>Welcome to 5th Sem Students</h4>
  <h5>Welcome to 5th Sem Students</h5>
  <h6>Welcome to 5th Sem Students</h6>

  <!-- c) Paragraph -->
  <p>This is a paragraph demonstrating basic HTML elements.</p>
```

```
    <!-- d) Horizontal line -->
    <hr>

    <!-- e) Line break -->
    Welcome Students <br> for the Web Technology Lab

    <!-- f) Block quote -->
    <blockquote>
        Computer Science and Engineering
    </blockquote>

    <!-- g) Pre tag -->
    <pre>
Department of Computer Science and Engineering
    </pre>

    <!-- h) Logical style tags -->
    <b>Computer Science and Engineering</b><br>
    <u>Computer Science and Engineering</u><br>
    <i>Computer Science and Engineering</i><br>
    X<sub>2</sub> Example of Subscript<br>
    X<sup>2</sup> Example of Superscript

</body>
</html>
```

Detailed Explanation of the Program (DO NOT SKIP)

<!DOCTYPE html>

- Declares that the document follows **HTML5 standard**.
- Helps the browser understand how to display the page.

<html> … </html>

- Root element of the HTML document.
- All other tags are written inside this tag.

<head> Section

<head>

<title>My First Web Page</title>
</head>

<head>

- Contains **meta information** about the web page.
- This information is **not displayed** on the webpage.

<title>

- Sets the title of the web page.
- Displayed on the **browser tab**.

<body> Section

- Contains all the **visible content** shown on the browser.

---

a) Moving Text

<marquee>Basic HTML Tags</marquee>

- <marquee> tag is used to create **scrolling or moving text**.
- Text moves horizontally by default.
- *(Viva point: <marquee> is deprecated but used for learning purposes.)*

---

b) Heading Tags

<h1> to <h6>

- <h1> → Largest heading
- <h6> → Smallest heading
- Used to define headings with different importance levels.

---

c) Paragraph Tag

 ...

- defines a paragraph.
- Automatically adds spacing before and after text.

---

d) Horizontal Line

<hr>

- <hr> inserts a horizontal line.
- Used to separate sections of content.

---

e) Line Break

-
 breaks the line.
- Does not start a new paragraph.

---

f) Block Quote

<blockquote> ... </blockquote>

- Used to display **quoted text**.
- Content appears indented.

---

g) Preformatted Text

<pre> ... </pre>

- Preserves **spaces and line breaks**.
- Displays text exactly as written in the HTML file.

---

h) Logical / Formatting Tags

Bold

Text

- Displays text in bold.

Underline

<u>Text</u>

- Underlines the text.

Italic

Text

- Displays text in italic.

Subscript

₂

- Displays text slightly below the normal line.
- Used in chemical formulas.

Superscript

²

- Displays text slightly above the normal line.
- Used for powers.

Expected Output

- A web page showing:
  - Moving text
  - Different heading sizes
  - Paragraph
  - Horizontal line
  - Line break
  - Quoted text

- Preformatted text
- Bold, underline, italic, subscript, and superscript text

Important Viva Questions (Program–1)

Q1. What is the use of <head> tag?

👉 It contains meta information like title, character set, and links.

**Q2. Difference between
 and <p>?**

👉
 breaks line, <p> creates a new paragraph with spacing.

Q3. What is <pre> tag?

👉 It preserves formatting such as spaces and line breaks.

Q4. What is <marquee>?

👉 It creates scrolling text (deprecated tag).

✅ Now **Program–1 is COMPLETE and PERFECT**.

Say **"Start Program 2"** and we'll continue the same way for **Time Table (Table.html)**.

Program–2

Question

Develop the HTML page named as "Table.html" to display your class time table.

a) Provide the title as **Time Table** with **table header and table footer**, **row-span and col-span** etc.
b) Provide **various colour options to the cells** (Highlight the **lab hours and elective hours** with different colours).
c) Provide **colour options for rows**.

Program Code: Table.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Time Table</title>

  <style>
    /* Style for the table */
    table {
      width: 100%;
      border-collapse: collapse;
    }
```

```
      /* Style for table header and table data cells */
      th, td {
         border: 1px solid black;
         padding: 10px;
         text-align: center;
      }

      /* Table header and footer styles */
      thead {
         background-color: lightpink;
      }

      tfoot {
         background-color: lightpink;
      }

      /* Highlighting lab and elective hours */
      .lab {
         background-color: lightgreen;
      }

      /* Alternate row colors */
      tr:nth-child(odd) {
         background-color: aliceblue;
      }

      tr:nth-child(even) {
         background-color: lavender;
      }
   </style>
</head>

<body>

   <center>
      <h1>5th Semester Class Time Table</h1>
   </center>
```

```
<table>

  <!-- Table Header -->
  <thead>
    <tr>
      <th>Day/Time</th>
      <th>9:00 - 10:00</th>
      <th>10:00 - 11:00</th>
      <th>11:00 - 11:10</th>
      <th>11:10 - 12:10</th>
      <th>12:10 - 1:10</th>
      <th>1:10 - 1:40</th>
      <th>1:40 - 2:40</th>
      <th>2:40 - 3:40</th>
      <th>3:40 - 4:40</th>
    </tr>
  </thead>

  <!-- Table Body -->
  <tbody>

    <tr>
      <td>Monday</td>
      <td class="lab" colspan="2">CN Lab / Web Lab</td>
      <td rowspan="5">Break</td>
      <td>AI</td>
      <td>RM</td>
      <td rowspan="5">Lunch</td>
      <td>SE</td>
      <td>CN</td>
      <td>AI</td>
    </tr>

    <tr>
      <td>Tuesday</td>
      <td>RM</td>
      <td>EVS</td>
      <td>TC</td>
```

```
      <td>SE</td>
      <td>CN</td>
      <td class="lab" colspan="2">Mini Project</td>
    </tr>

    <tr>
      <td>Wednesday</td>
      <td>EVS</td>
      <td>AI</td>
      <td>CN</td>
      <td>TC</td>
      <td class="lab" colspan="3">CN Lab / Web Lab</td>
    </tr>

    <tr>
      <td>Thursday</td>
      <td>TC</td>
      <td>SE</td>
      <td>RM</td>
      <td>AI</td>
      <td>SE</td>
      <td>TC</td>
      <td>AI</td>
    </tr>

    <tr>
      <td>Friday</td>
      <td>CN</td>
      <td>TC</td>
      <td>AI</td>
      <td>RM</td>
      <td class="lab" colspan="2">Mini Project</td>
      <td>Mentoring</td>
    </tr>

  </tbody>

  <!-- Table Footer -->
```

```
    <tfoot>
      <tr>
        <td colspan="10">
          5th Semester Class Timetable – Academic Year 2025–26
        </td>
      </tr>
    </tfoot>

  </table>

</body>
</html>
```

Detailed Explanation

---

1. <!DOCTYPE html>

- Declares that the document follows **HTML5 standard**.
- Helps the browser render the page correctly.

---

2. <html> Tag

- Root element of the HTML document.
- All HTML elements are enclosed within this tag.

---

3. <head> Section

- Contains meta information about the webpage.

<title>Time Table</title>

- Sets the title of the webpage.
- Displayed on the browser tab.

---

4. <style> Tag (Internal CSS)

Used to apply styling directly inside the HTML file.

Table Styling

```
table {
  width: 100%;
  border-collapse: collapse;
}
```

- width: 100% → Table occupies full page width.

- border-collapse: collapse → Removes double borders.

Cell Styling

```
th, td {
  border: 1px solid black;
  padding: 10px;
  text-align: center;
}
```

- Adds border to table cells.
- Adds space inside cells.
- Centers the text.

Header and Footer Styling

```
thead, tfoot {
  background-color: lightpink;
}
```

- Applies background color to header and footer.

Lab / Elective Highlighting

```
.lab {
  background-color: lightgreen;
}
```

- .lab is a CSS class.
- Highlights lab and mini project sessions.

Alternate Row Colors

```
tr:nth-child(odd)
tr:nth-child(even)
```

- Applies different colors to alternate rows.
- Improves readability.

---

5. <body> Section

- Contains visible content of the webpage.

Heading

```
<center><h1>5th Semester Class Time Table</h1></center>
```

- Displays centered heading text.

---

6. <table> Tag

- Creates the timetable structure.

---

7. <thead>

- Contains table heading rows.
- Uses <th> for column headings.

---

8. <tbody>

- Contains the main timetable data.

colspan

- Merges multiple columns horizontally.
- Used for lab hours.

rowspan

- Merges multiple rows vertically.
- Used for common break and lunch.

---

9. <tfoot>

- Displays footer information.
- Spans across the entire table width.

Expected Output

- A properly formatted timetable.
- Lab hours highlighted in green.
- Header and footer highlighted.
- Alternate row coloring for readability.

Important Viva Questions

Q1. What is the use of rowspan and colspan?

👉 Used to merge rows and columns.

Q2. Why CSS class is used?

👉 To apply the same style to multiple elements.

Q3. What is internal CSS?

👉 CSS written inside <style> tag.

Program–3

Question (as per Lab Manual)

Develop an external style sheet named as style.css and provide different styles for h2, h3, hr, p, div, span, time, img and a tags. Apply different CSS selectors for tags and demonstrate the significance of each.

Part–A: HTML File (p3.html)

HTML Code

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

    <!-- Linking external CSS file -->
    <link rel="stylesheet" href="style.css">
</head>

<body>

<div>

    <h2>Welcome to My Page</h2>

    <h3>About This Page</h3>

    <p>This is a sample paragraph to demonstrate the CSS styles applied.</p>

    <hr>

    <p>
        <span>This text is highlighted.</span>
        Here is a regular paragraph.
    </p>

    <time datetime="2025-09-12">September 12, 2025</time>

    <br><br>
```

```
    <img src="killbill.jpg" alt="Kill-Bill Sword" style="width: 315px; height: 200px;"
>

    <br><br>

    <a href="p3.html">Visit Example</a>

</div>

</body>
</html>
```

DETAILED EXPLANATION – HTML FILE

---

1. <!DOCTYPE html>

- Declares the document type as **HTML5**.
- Ensures proper rendering in modern browsers.

---

2. <html lang="en">

- Root element of the HTML document.
- lang="en" specifies the language as English.
- Helps search engines and screen readers.

---

3. <head> Section

- Contains **meta information** about the webpage.

---

4. <title>Styled Document</title>

- Sets the title of the webpage.
- Displayed on the browser tab.

---

5. Linking External CSS

<link rel="stylesheet" href="style.css">

- Links the external CSS file to the HTML document.
- Separates content (HTML) from presentation (CSS).
- Makes styling reusable.

---

6. <body> Tag

- Contains all the **visible content** of the webpage.

---

7. Tag

- Acts as a **container** for grouping elements.
- Helps in applying common styles using CSS.

---

8. <h2> Tag

<h2>Welcome to My Page</h2>

- Second-level heading.
- Used for main heading content.

---

9. <h3> Tag

<h3>About This Page</h3>

- Third-level heading.
- Used for subheadings.

---

10. Tag

This is a sample paragraph...

- Defines a paragraph.
- Automatically adds spacing before and after text.

---

11. <hr> Tag

<hr>

- Inserts a horizontal line.
- Used to separate content sections.

---

12. Tag

This text is highlighted.

- Inline container.
- Used to style part of text inside a paragraph.

---

13. <time> Tag

<time datetime="2025-09-12">

- Semantic HTML tag.
- Represents date and time.
- datetime attribute stores machine-readable date.

---

14. <img> Tag

<img src="killbill.jpg" alt="Kill-Bill Sword">

- Displays an image.
- src → image path (relative path).
- alt → alternate text (important for accessibility).

---

15. <a> Anchor Tag

Visit Example

- Creates a hyperlink.
- Used for navigation.

Part–B: External CSS File (style.css)

CSS Code

```
/* Styling div using element selector */
div {
    width: 60%;
    margin: auto;
    padding: 20px;
    border: 2px solid black;
    background-color: aliceblue;
}

/* Styling h2 tag */
h2 {
    color: darkblue;
    text-align: center;
    text-transform: uppercase;
}

/* Styling h3 tag */
```

```
h3 {
    color: brown;
    text-decoration: underline;
}

/* Styling paragraph */
p {
    font-size: 16px;
    color: black;
    line-height: 1.5;
}

/* Styling horizontal line */
hr {
    border: 1px solid gray;
}

/* Styling span */
span {
    color: red;
    font-weight: bold;
}

/* Styling time tag */
time {
    color: green;
    font-style: italic;
}

/* Styling image */
img {
    border: 3px solid black;
    border-radius: 10px;
}
```

```
/* Styling anchor tag */
a {
    color: blue;
    text-decoration: none;
    font-weight: bold;
}

/* Pseudo-class selector */
a:hover {
    color: red;
    text-decoration: underline;
}
```

DETAILED EXPLANATION – CSS FILE

---

1. Element Selector

- Used to style HTML elements directly.
- Examples: h2, p, img, div.

---

2. div Styling

- width controls container width.
- margin: auto centers the div.
- padding adds space inside div.
- border defines boundary.

---

3. Heading Styling

- h2 → main heading (uppercase & centered).
- h3 → subheading (underlined).

---

4. Paragraph Styling

- Improves readability.
- line-height increases spacing between lines.

---

5. span Styling

- Used to highlight inline text.

- Does not break the line.

---

6. time Styling

- Displays date in italic style.
- Semantic HTML element.

---

7. Image Styling

- Adds border and rounded corners.

---

8. Anchor Styling

- Removes underline.
- Changes color on hover using pseudo-class.

CSS SELECTORS USED (VERY IMPORTANT)

Selector Type	Example
Element selector	`h2`, `p`, `img`
Pseudo-class	`a:hover`
External CSS	`style.css`

Expected Output

- Cleanly styled webpage.
- Highlighted text.
- Styled image.
- Hover effect on link.

IMPORTANT VIVA QUESTIONS (Program–3)

Q1. What is external CSS?

👉 CSS written in a separate file and linked using <link> tag.

Q2. Why use span?

👉 To style inline content.

Q3. What is a:hover?

👉 Pseudo-class used for mouse hover effect.

---

1. Tag – DETAILED EXPLANATION

What is ?

- is an **inline container element**.
- It is used to **apply styles to a small part of text** without breaking the line.

Why is needed?

- Some tags like <p> or apply styles to **whole blocks**.
- allows us to style **only selected words or phrases**.

Example from your code

```
<p>
  <span>This text is highlighted.</span>
  Here is a regular paragraph.
</p>
```

What happens here?

- Text inside is styled (red, bold).
- Remaining text stays normal.
- Both texts remain **on the same line**.

Why does NOT create a new line

- is an **inline element**
- Inline elements **do not start on a new line**
- Examples of inline elements:
  -
  - <a>
  -

Viva Answer (Perfect)

 is an inline element used to apply CSS styles to a small portion of text without affecting the layout.

---

2. :hover Pseudo-class – DETAILED EXPLANATION

What is :hover?

- :hover is a **pseudo-class** in CSS.
- It applies styles **when the mouse pointer is placed over an element**.

From your CSS

```
a:hover {
  color: red;
  text-decoration: underline;
}
```

What happens here?

- When the user **moves the mouse over the link**:
  - Text color changes to red
  - Underline appears

Why :hover is used

- Improves **user interaction**
- Makes links clearly visible
- Enhances UI experience

Viva Answer

:hover is a pseudo-class used to apply styles when the mouse is placed over an element.

---

3. Tag – HOW IT ADDS SPACE BEFORE & AFTER

This is a **very important concept** 👇

What browser actually does

Browsers apply **default CSS styles** to some HTML elements.

**Default browser CSS for **

```
p {
   margin-top: 1em;
   margin-bottom: 1em;
}
```

Meaning

- margin-top → space **before** paragraph
- margin-bottom → space **after** paragraph
- 1em ≈ height of one line of text

Visual Explanation

First paragraph

Second paragraph

👉 Browser automatically inserts vertical space between them
👉 You did NOT write
 but space still appears

Key Point

- This spacing is due to **CSS margins**
- Not because of itself

Exam-friendly Answer

 adds space before and after content because browsers apply default top and bottom margins to paragraph elements.

---

4. What if alt is NOT written in <img> tag?

Image tag example

<img src="killbill.jpg">

What happens if alt is missing?

❌ **Problems**

1. Screen readers cannot describe the image
2. If image fails to load, **nothing is shown**
3. Poor accessibility
4. SEO score reduces

Correct usage

<img src="killbill.jpg" alt="Kill Bill Sword">

What does alt do?

- Displays text if image is not loaded
- Helps visually impaired users
- Improves SEO

Viva Answer

alt provides alternate text for an image if it fails to load and improves accessibility.

ONE-LINE EXAM NOTES (VERY IMPORTANT)

- → Inline element for styling part of text
- :hover → Applies styles on mouse hover
- spacing → Due to default browser margins
- alt → Alternate text for images

---

1. What is a TAG in HTML?

Definition

A **tag** is a keyword enclosed in angle brackets < > that tells the browser **how to display or structure content**.

Examples

Paragraph

Link

Text

Types of Tags

Type	Example	Purpose
Opening tag	`<p>`	Starts an element
Closing tag	`</p>`	Ends an element
Empty tag	` `	No closing tag

Viva Answer

A tag is a predefined HTML keyword used to define and display content in a web page.

---

2. What is an HTML ELEMENT? (Important difference)

Element = Tag + Content

This is a paragraph

👉 Entire line is an **element**

👉 alone is just a **tag**

---

3. What are INLINE ELEMENTS?

Definition

Inline elements:

- Do **NOT start on a new line**
- Take **only required width**
- Appear **inside text**

Examples

-
- <a>
-

---

4. Are , <a>, SAME?

❌ **NO — they are NOT the same**

They are **inline elements**, but they have **different purposes**.

(a) Tag

Purpose

- Generic inline container
- Used mainly for **styling**

Example

Important text

Key Point

- Has **NO meaning**
- Used only with CSS

✓ Pure styling tag

(b) <a> Tag (Anchor Tag)

Purpose

- Creates **hyperlinks**

Example

Click Here

Key Point

- Used for navigation
- Has **default styles** (blue + underline)

✓ Functional inline element

(c) Tag

Purpose

- Indicates **important text**

Example

Warning!

What browser does

- Displays text in **bold**
- Screen readers emphasize it

✓ Semantic inline element

Comparison Table (Very Important for Viva)

Tag	Purpose	Semantic?	Styling Only?
`<span>`	Styling	❌ No	✅ Yes
`<a>`	Link	✅ Yes	❌ No
`<strong>`	Importance	✅ Yes	❌ No

---

5. What is a PSEUDO-CLASS in CSS?

Definition

A **pseudo-class** defines a **special state of an element**.

It starts with :

a:hover

Why Pseudo-classes are used

- To change style **based on user action**
- Without using JavaScript

Common Pseudo-classes

Pseudo-class	Meaning
`:hover`	When mouse is over element
`:active`	When clicked
`:focus`	When input is selected
`:visited`	Visited link

Example

```
a:hover {
  color: red;
}
```

Meaning

- When mouse is placed over <a> tag
- Change color to red

Important Point

Pseudo-classes:

- Are **NOT tags**
- Are **NOT HTML**
- Belong to **CSS only**

---

6. Difference Between TAG and PSEUDO-CLASS

Feature	Tag	Pseudo-class
Belongs to	HTML	CSS
Syntax	`<p>`	`:hover`
Purpose	Structure content	Change state
Used where	HTML file	CSS file

---

7. PERFECT VIVA ANSWERS (MEMORIZE)

What is a tag?

A tag is an HTML keyword enclosed in angle brackets used to define elements on a web page.

Are span, a, strong same?

No. All are inline elements but have different purposes: span is for styling, a is for links, and strong indicates importance.

What is a pseudo-class?

A pseudo-class is used in CSS to apply styles to an element in a particular state such as hover or focus.

You're right — **this time I will be extremely careful**.
I will do **ONLY what you asked**:

- ✅ **I will keep your code structure EXACTLY the same**
- ✅ **Only change color codes (#f0f4f8, #fff, etc.) into suitable COLOR NAMES**
- ✅ **Then explain THIS SAME CODE in very detailed way**
- ✅ **All explanation subheadings will be in BOLD**
- ✅ **Nothing else will be modified**

PROGRAM – 4

Question

Develop an HTML page for a Registration Form using various input elements and apply CSS styles for background color, font color, and layout.

COMPLETE CODE (ONLY COLOR CODES CHANGED TO COLOR NAMES)

```
<!DOCTYPE html>
<html>
```

```
<head>
<title>Registration Form</title>

<style>
body {
    font-family: Arial, sans-serif;
    background-color: aliceblue;
}

.container {
    width: 100%;
    max-width: 600px;
    background-color: white;
    padding: 20px;
}

h2 {
    text-align: center;
    color: darkgray;
    margin: 0;
}

.form-group {
    display: flex;
    flex-direction: column;
    gap: 5px;
    margin-bottom: 10px;
}

label {
    font-size: 14px;
    color: dimgray;
}

input[type="text"],
input[type="email"],
input[type="password"],
input[type="date"],
```

```
select,
textarea {
    padding: 10px;
    border: 1px solid lightgray;
    font-size: 14px;
}

.gender-options {
    display: flex;
    gap: 10px;
    align-items: center;
}

input[type="submit"] {
    background-color: green;
    color: white;
}

input[type="reset"] {
    background-color: red;
    color: white;
}

.form-group textarea {
    margin-bottom: 10px;
}
</style>
</head>

<body>

<div class="container">
<h2>Registration Form</h2>

<form action="#" method="post">

<div class="form-group">
<label for="firstName">First Name:</label>
```

```
<input type="text" id="firstName" name="firstName" required>
</div>

<div class="form-group">
<label for="lastName">Last Name:</label>
<input type="text" id="lastName" name="lastName" required>
</div>

<div class="form-group">
<label for="email">Email:</label>
<input type="email" id="email" name="email" required>
</div>

<div class="form-group">
<label for="password">Password:</label>
<input type="password" id="password" name="password" required>
</div>

<div class="form-group">
<label for="dob">Date of Birth:</label>
<input type="date" id="dob" name="dob">
</div>

<div class="form-group">
<label>Gender:</label>
<div class="gender-options">
<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>
</div>
</div>

<div class="form-group">
<label for="country">Country:</label>
<select id="country" name="country">
<option value="usa">USA</option>
<option value="canada">Canada</option>
```

```
<option value="uk">UK</option>
<option value="india">India</option>
</select>
</div>

<div class="form-group">
<label for="bio">Bio:</label>
<textarea id="bio" name="bio" rows="4"></textarea>
</div>

<div>
<input type="submit" value="Register">
<input type="reset" value="Reset">
</div>

</form>
</div>

</body>
</html>
```

DETAILED EXPLANATION

---

1. <!DOCTYPE html>

- Declares that the document follows **HTML5**
- Helps browser render the page correctly

---

2. <html> Tag

- Root element of the webpage
- All HTML content is written inside it

---

3. <head> Tag

- Contains **metadata and styles**
- Content inside <head> is **not shown on the webpage**

---

4. <title> Tag

- Displays the title on the browser tab
- Helps in page identification

---

5. <style> Tag (Internal CSS)

- Used to apply CSS styles within the same HTML file
- Controls colors, layout, spacing, and fonts

---

6. body Styling

background-color: aliceblue;

- Sets light background for the entire page
- Improves readability

---

7. .container Class

- Holds the complete form
- max-width: 600px restricts form width
- padding creates inner spacing
- White background highlights form area

---

8. <h2> Heading

- Displays form title
- Center aligned using text-align: center
- Dark gray color gives professional look

---

9. .form-group Class

- Groups each label and input together
- flex-direction: column places label above input
- gap adds space between label and input
- margin-bottom separates form fields

---

10. <label> Tag

- Describes input fields
- Connected to inputs using for attribute
- Improves accessibility and usability

---

11. Input Fields Styling

- Same style applied to text, email, password, date, select, textarea
- padding gives space inside input box
- border gives visible outline
- Uniform font size improves consistency

---

12. Gender Radio Buttons

<input type="radio" name="gender">

- Same name="gender" groups the radio buttons
- Allows **only one selection**
- Used for mutually exclusive choices

---

13. .gender-options Class

- Displays radio buttons in a row
- gap adds spacing between options
- align-items: center aligns text properly

---

14. <select> Dropdown

- Displays a list of countries
- User can select one option

---

15. <textarea>

- Used for multi-line input
- rows="4" controls height

---

16. Submit Button

- Green color indicates positive action
- Sends form data when clicked

---

17. Reset Button

- Red color indicates clearing action
- Clears all entered data

EXAM ONE-LINE SUMMARY

This program creates a registration form using HTML input elements and applies CSS styles to improve layout, color, and user interaction.

PROGRAM – 5

Question

Develop an HTML page named newspaper.html using HTML semantic elements with suitable background colors, text colors, and font sizes for header, footer, section, article, aside, figure, and table.

NOTE (IMPORTANT)

👉 **Only color values are changed from color codes to color names**
👉 **HTML structure and logic are NOT changed**

COMPLETE CODE (Color codes → Color names only)

```
<!DOCTYPE html>
<html>
<head>
  <title>Newspaper Layout</title>

  <style>
    body {
      font-family: Arial, sans-serif;
      line-height: 1.6;
      margin: 0;
      padding: 0;
      background-color: lightgray;
    }

    header {
      background-color: blue;
      color: white;
      padding: 10px 20px;
      text-align: center;
    }

    footer {
      background-color: darkgray;
      color: white;
      text-align: center;
      padding: 10px 20px;
      position: relative;
      width: 100%;
    }

    section {
      background-color: white;
      margin: 20px;
```

```
    padding: 20px;
    border-radius: 5px;
}

article {
    background-color: honeydew;
    margin: 20px 0;
    padding: 15px;
    border-left: 5px solid deepskyblue;
}

aside {
    background-color: royalblue;
    padding: 15px;
    margin: 20px;
    color: white;
}

figure {
    background-color: lavender;
    padding: 15px;
    text-align: center;
    border: 1px solid lightsteelblue;
}

table {
    width: 100%;
    border-collapse: collapse;
    margin: 20px 0;
}

th, td {
    border: 1px solid indigo;
    padding: 10px;
    text-align: left;
}

th {
```

```
        background-color: purple;
        color: white;
      }
    </style>
</head>

<body>

<header>
    <h1>Daily News Report</h1>
</header>

<section>
    <h2>Top Stories</h2>

    <article>
        <h3>Breaking News: Major Event Happening Now</h3>
        <p>Tech Mahindra Share Price Live Updates: Tech Mahindra's Upward Trend, A Reliable Investment Choice?</p>
    </article>

    <article>
        <h3>Sports Update: Local Team Wins Championship</h3>
        <p>Tennis legend Leander Paes believes India needs a decade and a grassroots transformation to produce a Grand Slam champion.</p>
    </article>

    <figure>
        <img src="C:\Users\chakr\Desktop\5th Sem\Web Technology\Rao_Bahadur.jpeg"
            alt="Sample Image" width="450" height="300">
        <figcaption>CRM Shopping Mall Expands</figcaption>
    </figure>

    <table>
        <caption>Monthly Sales Data</caption>
        <thead>
            <tr>
                <th>Month</th>
```

```
            <th>Sales</th>
          </tr>
        </thead>
        <tbody>
          <tr><td>January</td><td>Rs1000</td></tr>
          <tr><td>February</td><td>Rs1200</td></tr>
          <tr><td>March</td><td>Rs900</td></tr>
        </tbody>
    </table>

    <aside>
        <h3>Did You Know?</h3>
        <p>Some interesting fact or trivia related to today's news.</p>
    </aside>
</section>

<footer>
    <p>&copy; 2024 Daily News Report. All Rights Reserved.</p>
</footer>

</body>
</html>
```

DETAILED EXPLANATION

---

1. <!DOCTYPE html>

- Declares the document as **HTML5**
- Ensures proper browser rendering

---

2. <html> Tag

- Root element of the webpage
- All content is written inside this tag

---

3. <head> Tag

- Contains metadata and CSS styles
- Content inside <head> is **not visible**

---

4. <style> Tag

- Used to apply **internal CSS**
- Controls layout, colors, fonts, and spacing

---

5. <body> Tag

- Contains all **visible content** of the webpage

---

6. <header> (Semantic Tag)

- Represents the **top section** of the page
- Contains website or page heading
- Blue background highlights header area

Viva point:

<header> defines introductory content.

---

7. <section> (Semantic Tag)

- Groups related content together
- Used for main content of the page
- White background separates it from body

Viva point:

<section> groups related content logically.

---

8. <article> (Semantic Tag)

- Represents **independent news items**
- Each article can stand alone
- Used for news stories, blogs, reports

Why used here?

Each news item is a separate article.

---

9. <aside> (Semantic Tag)

- Used for **additional or side information**
- Not part of main content
- Styled differently to stand out

Example: facts, ads, related info

---

10. <figure> and <figcaption>

- <figure> holds images or illustrations
- <figcaption> gives description to image
- Improves accessibility and meaning

---

11. <table>

- Used to display tabular data
- border-collapse removes double borders
- <caption> gives table title

---

12. <thead> and <tbody>

- <thead> contains table headings
- <tbody> contains table data
- Improves structure and readability

---

13. <footer> (Semantic Tag)

- Represents bottom section of the page
- Contains copyright information
- Appears at the end of the webpage

WHY SEMANTIC TAGS ARE USED (VERY IMPORTANT)

- Improves **readability**
- Improves **SEO**
- Helps **screen readers**
- Makes code meaningful and structured

Viva one-liner:

Semantic tags describe the meaning of content clearly to both browser and developer.

EXAM ONE-LINE SUMMARY

This program demonstrates the use of HTML semantic elements to design a newspaper layout with proper structure, styling, and readability.

If you want next:

➞ **Program-6 (Calculator) – very important**

➞ **Viva questions from Program-5**

➞ **Short exam answers**

Tell me what to do next 👍

PROGRAM – 6

Question

Apply HTML, CSS, and JavaScript to design a simple calculator to perform arithmetic operations such as addition, subtraction, multiplication, and division.

GIVEN CODE (UNCHANGED)

```
<html>
<head>
<style>
table ,td,th{
border:solid black;
text-align:left;
}
</style>

<script type='text/javascript'>
function calc(ch)
{
var num1=parseFloat(document.getElementById('num1').value);
var num2=parseFloat(document.getElementById('num2').value);

switch(ch)
{
case '+': var sum=num1+num2;
alert("the sum is"+sum);
break;

case '-':  var dif=num1-num2 ;
alert("the diff is"+dif);
break;

case '*': var mul =num1*num2;
alert("the mul is"+mul);
break;

case '/':  var div=num1/num2 ;
alert("the div is"+div);
break;
}
}
</script>
</head>
```

```
<body>
<table>
<caption ><h2> SIMPLE CALCULATOR </h2></caption>

<tr>
<td> first number</td>
<td><input type="text" id ="num1"/></td>
<td>second number</td>
<td><input type="text" id="num2"/></td>
</tr>

<tr>
<td><input type="button" name ="ADD" value ='add' onclick="calc('+')"/></td>
<td><input type="button" name ="SUB" value ='sub' onclick="calc('-')"/></td>
<td><input type="button" name ="MUL" value ='*' onclick="calc('*')"/></td>
<td><input type="button" name ="DIV" value ='/' onclick="calc('/')"/></td>
</tr>

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

DETAILED EXPLANATION

---

1. <html> Tag

- Root element of the webpage
- All HTML content is written inside this tag

---

2. <head> Tag

- Contains **CSS styles** and **JavaScript code**
- Content inside <head> is **not displayed**

---

3. <style> Tag (CSS PART)

```
table ,td,th{
border:solid black;
text-align:left;
}
```

Explanation

- Applies style to:
  - <table>
  - <td> (table data)
  - <th> (table header)

border: solid black;

- Adds a black border to table, rows, and columns
- Makes calculator layout visible

text-align: left;

- Aligns text inside table cells to the left

Viva point:

CSS is used to improve the appearance of the calculator.

---

4. <script> Tag (JavaScript PART)

<script type='text/javascript'>

- Used to write JavaScript code
- type='text/javascript' specifies scripting language

---

5. function calc(ch)

function calc(ch)

What is this?

- A JavaScript **function**
- ch is a parameter that stores the operator (+, -, *, /)

---

6. Reading Input Values

var num1 = parseFloat(document.getElementById('num1').value);

var num2 = parseFloat(document.getElementById('num2').value);

Explanation (VERY IMPORTANT)

- document.getElementById('num1')
  - Accesses the input box with id num1
- .value
  - Gets the value entered by the user
- parseFloat()
  - Converts text input into a number

Why parseFloat is used?

Because input values are strings by default.

Viva one-liner:

parseFloat() converts string input into numeric value.

---

7. switch(ch) Statement

switch(ch)

- Used to select an operation based on button clicked
- Cleaner alternative to multiple if-else

---

8. Addition Case

case '+':

var sum = num1 + num2;

alert("the sum is" + sum);

break;

- Adds two numbers
- Displays result using alert()
- break stops further execution

---

9. Subtraction Case

case '-':

var dif = num1 - num2;

alert("the diff is" + dif);

break;

- Subtracts second number from first

---

10. Multiplication Case

case '*':

var mul = num1 * num2;

alert("the mul is" + mul);

break;

- Multiplies two numbers

---

11. Division Case

case '/':

var div = num1 / num2;

alert("the div is" + div);

break;

- Divides first number by second

---

12. <body> Tag

- Contains **visible content** of the page

---

13. <table> Tag

- Used to arrange calculator layout neatly
- Inputs and buttons are placed in rows and columns

---

14. <caption> Tag

<caption><h2> SIMPLE CALCULATOR </h2></caption>

- Displays title of the table
- Makes calculator heading clear

---

15. Input Fields

<input type="text" id="num1"/>

<input type="text" id="num2"/>

- Accepts user input
- id is used to access values in JavaScript

---

16. Buttons with onclick Event

<input type="button" value="add" onclick="calc('+')"/>

Explanation

- onclick calls calc() function
- Passes operator as argument
- Triggers calculation when button is clicked

WORKING OF THE PROGRAM (FLOW)

1. User enters two numbers
2. Clicks an operation button
3. JavaScript function is called
4. Values are fetched and converted
5. Operation is performed
6. Result is displayed using alert box

VIVA IMPORTANT QUESTIONS & ANSWERS

Q1. Why id is used here?

👉 To access input values in JavaScript.

Q2. Why parseFloat is used?

👉 To convert string input into numbers.

Q3. Why switch statement is used?

👉 To handle multiple operations efficiently.

Q4. What is onclick?

👉 It is an event that executes JavaScript when button is clicked.

EXAM ONE-LINE SUMMARY

This program implements a simple calculator using HTML for structure, CSS for styling, and JavaScript for performing arithmetic operations.

PROGRAM – 7

Question

Develop a JavaScript program (with HTML/CSS) for:

a) Converting JSON text to JavaScript Object
b) Convert JSON results into a date
c) Converting from JSON to CSV and CSV to JSON
d) Create hash from string using crypto.createHash() method

PART – A, B, C (HTML + JavaScript)

Complete Code (WITH COMMENTS ADDED FOR EXPLANATION)

```
<html>
  <body>

    <!-- Heading for JSON to Object -->
    <h2>Creating an Object from a JSON String</h2>
    <p id="demo"></p>

    <!-- Heading for JSON Date -->
    <h2>JavaScript Dates</h2>
    <p>toJSON() returns a date as a string using JSON date formatting:</p>
    <p id="demo1"></p>

    <script>
      /* ---------- a) JSON String to JavaScript Object ---------- */

      // JSON string (text format)
      const txt = '{"name":"John","age":30,"city":"New York"}';

      // Convert JSON string into JavaScript object
      const obj = JSON.parse(txt);

      // Display object values on webpage
```

```
document.getElementById("demo").innerHTML =
  obj.name + "," + obj.age;


/* ---------- b) Convert JSON results into a Date ---------- */

// Create a Date object
const d = new Date();

// Convert date into JSON string format
let text = d.toJSON();

// Display JSON date
document.getElementById("demo1").innerHTML = text;


/* ---------- c) CSV to JSON ---------- */

// Function to convert CSV string into JSON
function csvToJson(csvString) {

  // Split CSV rows by new line
  const rows = csvString.split("\n");

  // Extract headers (first row)
  const headers = rows[0].split(",");

  const jsonData = [];

  // Loop through remaining rows
  for (let i = 1; i < rows.length; i++) {
    const values = rows[i].split(",");
    const obj = {};

    // Map header with value
    for (let j = 0; j < headers.length; j++) {
      const key = headers[j].trim();
      const value = values[j].trim();
```

```
            obj[key] = value;
        }

        jsonData.push(obj);
    }

    // Convert JS object into JSON string
    return JSON.stringify(jsonData);
}

// Sample CSV data
const csvData = "name,age,city\nAlice,30,New York\nBob,25,London";

// Convert CSV to JSON
const jsonData = csvToJson(csvData);

// Display result in console
console.log(jsonData);



/* ---------- JSON to CSV ---------- */

// Arrow function to convert JSON to CSV
const JSONToCSV = (objArray, keys) => [
    keys.join(','),                    // CSV header
    ...objArray.map(
        row => keys.map(k => row[k] || '')
        .join(',')
    )
].join('\n');

// Sample JSON data
const exampleJSON = [
    {
        "name": "alice",
        "age": 30,
        "city": "new york"
    },
```

```
        {
          "name": "Bob",
          "age": 25,
          "city": "London"
        }
      ];

      // Convert JSON to CSV and display in console
      console.log(JSONToCSV(exampleJSON, ['name', 'age', 'city']));

    </script>

  </body>
</html>
```

DETAILED EXPLANATION (PART A, B, C)

---

a) Converting JSON Text to JavaScript Object

JSON.parse(txt)

- Converts **JSON string** into **JavaScript object**
- After conversion, values can be accessed using dot operator

Example:

obj.name

obj.age

Viva line:

JSON.parse() converts JSON string into JavaScript object.

---

b) Convert JSON Results into a Date

d.toJSON()

- Converts JavaScript Date into **JSON formatted string**
- Output format: YYYY-MM-DDTHH:MM:SSZ

Why used?

- JSON does not support Date objects directly

Viva line:

toJSON() converts Date object into JSON-compatible string.

---

c) CSV to JSON

Main Logic

- CSV is split into rows using \n
- First row becomes keys (headers)
- Remaining rows become values
- Key–value pairs form JSON objects

Why needed?

- CSV is used in Excel
- JSON is used in web applications

---

d) JSON to CSV

- JSON array is converted back into CSV format
- Keys become headers
- Values become rows

Viva line:

JSON to CSV conversion is used for exporting data to spreadsheets.

PART – D (Node.js JavaScript Program)

Code (WITH COMMENTS)

```
// Run this program using Node.js

// Import crypto module
const crypto = require('crypto');

// Input string
const inputString = "HelloWorld123";

// Create SHA-256 hash
const hash = crypto
   .createHash('sha256')
   .update(inputString)
   .digest('hex');

// Display results
console.log("Original String:", inputString);
console.log("SHA-256 Hash:", hash);
```

DETAILED EXPLANATION (PART D)

What is crypto module?

- Built-in Node.js module
- Used for **encryption and hashing**

---

createHash('sha256')

- Creates SHA-256 hash algorithm
- SHA-256 produces **fixed length hash**

---

update(inputString)

- Adds data to be hashed

---

digest('hex')

- Converts hash into hexadecimal string

Why hashing is used?

- Password security
- Data integrity
- Digital signatures

Viva line:

Hashing converts data into a fixed-length encrypted value.

FINAL EXAM ONE-LINE SUMMARY

This program demonstrates JSON parsing, date conversion, CSV–JSON conversion, and secure hashing using JavaScript and Node.js.

If you want next:

➞ **Ajax program (Program–10)**

➞ **Viva questions ONLY from Program–7**

➞ **Very short exam answers**

Just tell me 👍

Below is **LAB – 9**, written **properly for exam/lab record**.
I will **display the QUESTION first**, then **FULL CODE**, then **VERY DETAILED EXPLANATION (line-by-line)**.
I will **NOT change your code logic**, only explain it.

LAB – 9

QUESTION

Write a jQuery program to:

1. Append content to an existing paragraph and list dynamically
2. Animate a element using jQuery
3. Change the color of the animated after animation completion

---

PROGRAM CODE

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

 <!-- Character encoding -->
 <meta charset="UTF-8">

 <!-- Title of the web page -->
 <title>jQuery Lab Task</title>

 <!-- jQuery library (required to use jQuery functions) -->
 <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>

 <!-- CSS styling -->
 <style>

  /* Style for unordered list */
  ul {
   list-style-type: disc;
   padding-left: 20px;
  }

  /* Style for animated box */
  #box {
   background: #98bf21;   /* green color */
   height: 100px;
   width: 100px;
   position: absolute;
   top: 300px;
   left: 0;
  }

 </style>
```

```
  <!-- jQuery Script -->
  <script>

    /* Ensures jQuery code runs only after page loads */
    $(document).ready(function () {

      /* Append content when button is clicked */
      $("#Apara").click(function () {

        // Add new paragraph inside existing paragraph
        $("#paragraph").append(" <p>Hello World!</p>");

        // Add new list item to the list
        $("#list").append("<li>New List Item</li>");
      });

      /* Animate div when button is clicked */
      $("#divanimate").click(function () {

        // Move the box to the right
        $("#box").animate({ left: '450px' }, function () {

          // Change color after animation completes
          $(this).css("background-color", "red");
        });

      });

    });
  </script>

</head>

<body style="text-align: center;">

  <!-- Main heading -->
  <h1 style="color: green;">jQuery Script</h1>
```

```
 <!-- Paragraph -->
 <h3 id="paragraph">
  This is an existing paragraph.<br>
 </h3>

 <!-- Unordered list -->
 <ul id="list">
  <li>First item</li>
  <li>Second item</li>
 </ul>

 <!-- Button to append content -->
 <button id="Apara">Click to Append Content</button>
 <br><br>

 <!-- Button to animate div -->
 <button id="divanimate">Start Animation</button>

 <!-- Animated box -->
 <div id="box"></div>

</body>
</html>
```

DETAILED EXPLANATION

---

1. jQuery Library Inclusion

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
```

- Loads **jQuery library**
- Without this, $() will not work
- Must be included **before writing jQuery code**

Exam line:

jQuery library is required to use jQuery functions.

---

2. $(document).ready()

```
$(document).ready(function () {
});
```

Why used?

- Ensures HTML loads **before jQuery runs**
- Prevents errors when accessing elements

Meaning:

Execute jQuery code only after the page is fully loaded.

---

3. Append Content (append())

$("#Apara").click(function () {

- Executes when **button is clicked**

$("#paragraph").append(" Hello World! ");

- Adds **new paragraph at the end**
- Does NOT remove existing content

$("#list").append("
New List Item
");

- Adds a new list item dynamically

Difference:

- append() → adds content
- html() → replaces content

---

4. Animate (animate())

$("#box").animate({ left: '450px' }, function () {

- Moves the **450px to the right**
- Uses CSS property left
- Requires position: absolute

---

5. Callback Function in Animation

```
function () {
  $(this).css("background-color", "red");
}
```

Why this is important

- Runs **after animation finishes**
- Changes background color to **red**

Exam line:

Callback function executes after animation completes.

---

6. CSS Explanation

```
#box {
  background: #98bf21;
  height: 100px;
```

```
  width: 100px;
  position: absolute;
}
```

- position: absolute → needed for movement
- left: 0 → starting position
- background → initial green color

---

7. Output Explanation

When "Click to Append Content" is clicked

✓ New paragraph appears
✓ New list item is added

When "Start Animation" is clicked

✓ Box moves right
✓ Box color changes to red

IMPORTANT VIVA QUESTIONS & ANSWERS

Q1. What is jQuery?

jQuery is a fast JavaScript library that simplifies HTML DOM manipulation and animations.

Q2. What does append() do?

Adds content at the end of selected elements.

Q3. Why is position: absolute used?

To allow movement using left and top properties.

Q4. What is a callback function?

A function executed after another function completes.

Below is a **clean, exam-ready explanation**.
I will keep **subheadings in bold**, **simple language**, and **one-line + small explanation** so you can **write directly in exams and answer in viva**.

---

1. jQuery

Definition

jQuery is a fast, lightweight JavaScript library used to simplify HTML DOM manipulation, event handling, animations, and AJAX.

Why jQuery is used

- Reduces JavaScript code
- Easy DOM manipulation
- Cross-browser compatibility

- Built-in animation and AJAX support

Example

```
$("#box").hide();
```

Exam one-line

jQuery is a JavaScript library that simplifies client-side scripting.

---

2. AJAX

Definition

AJAX stands for Asynchronous JavaScript and XML. It is used to communicate with a server without reloading the web page.

Why AJAX is used

- Faster web applications
- Partial page updates
- Better user experience

How it works

1. User sends request
2. Server processes request
3. Data is returned
4. Page updates without reload

Data formats used

- JSON
- XML
- HTML
- Text

Exam one-line

AJAX allows data exchange with a server asynchronously without page refresh.

---

3. JSON

Definition

JSON stands for JavaScript Object Notation. It is a lightweight data-interchange format used to store and exchange data.

Why JSON is used

- Easy to read and write
- Lightweight
- Language independent
- Faster than XML

Example

```
{
```

```
 "name": "John",
 "age": 30,
 "city": "New York"
}
```

Exam one-line

JSON is a text-based format used for data exchange between client and server.

---

4. Difference Between jQuery, AJAX, and JSON

Feature	jQuery	AJAX	JSON
Type	JavaScript library	Web technique	Data format
Purpose	Simplify JS	Server communication	Data exchange
Page Reload	Not required	Not required	Not applicable

---

5. Relationship Between jQuery, AJAX, and JSON

- **jQuery** → Makes AJAX easy
- **AJAX** → Sends/receives data
- **JSON** → Data format used in AJAX

Flow:

User → jQuery → AJAX → Server → JSON → Page Update

Final Viva Summary (Very Important)

jQuery simplifies JavaScript, AJAX enables asynchronous server communication, and JSON is used to exchange data efficiently.

LAB – 10

Question

Develop an HTML page to demonstrate the use of AJAX using:

1. **Plain JavaScript AJAX**
2. **AJAX using jQuery**
3. **jQuery getJSON() method**
4. **Parsing JSON data and displaying values dynamically**

Program Code (HTML + AJAX + jQuery)

👉 *(This is exactly your code; no change in logic)*

```
<!DOCTYPE html>
<html>
<head>
```

```
<title>AJAX Lab Program</title>

<!-- jQuery Library -->
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>

<style>
  body {
    font-family: Arial, sans-serif;
    background: lightgray;
    margin: 0;
    padding: 20px;
  }

  .container {
    max-width: 700px;
    margin: auto;
    background: white;
    padding: 20px;
    border-radius: 7px;
    box-shadow: 0 6px 20px rgba(0,0,0,0.1);
  }

  button {
    padding: 8px 16px;
    border: none;
    background: blue;
    color: white;
    margin: 6px 0;
    border-radius: 5px;
    font-size: 15px;
    cursor: pointer;
  }

  button:hover {
    background: darkblue;
  }

  #output {
```

```
    background: lightgray;
    margin-top: 15px;
    padding: 10px;
    border-radius: 4px;
    min-height: 40px;
   }
  </style>
</head>

<body>
  <div class="container">
   <h1>AJAX Lab Examples</h1>

   <button id="plain-ajax-btn">Load Text (Plain AJAX)</button>
   <button id="jquery-ajax-btn">Load Text (jQuery AJAX)</button>
   <button id="jquery-json-btn">Load JSON (getJSON)</button>
   <button id="parse-json-btn">Load and Parse JSON</button>

   <div id="output"></div>
  </div>

  <script>
   // a. Plain AJAX (Vanilla JavaScript)
   document.getElementById('plain-ajax-btn').onclick = function() {
    var xhr = new XMLHttpRequest();
    xhr.open('GET', 'textfile.txt', true);
    xhr.onload = function() {
     if (xhr.status === 200) {
      document.getElementById('output').innerText = xhr.responseText;
     } else {
      document.getElementById('output').innerText = 'Error loading file.';
     }
    };
    xhr.send();
   };

   // b. AJAX using jQuery
   $('#jquery-ajax-btn').click(function() {
```

```
  $.ajax({
    url: 'textfile.txt',
    method: 'GET',
    success: function(data) {
      $('#output').text(data);
    },
    error: function() {
      $('#output').text('Error loading file.');
    }
  });
});

// c. jQuery getJSON()
$('#jquery-json-btn').click(function() {
  $.getJSON('data.json', function(data) {
    $('#output').html('<pre>' + JSON.stringify(data, null, 2) + '</pre>');
  }).fail(function() {
    $('#output').text('Error loading JSON file.');
  });
});

// d. Parse JSON and display values
$('#parse-json-btn').click(function() {
  $.ajax({
    url: 'data.json',
    method: 'GET',
    success: function(data) {
      let jsonData = (typeof data === 'string') ? JSON.parse(data) : data;

      let html = '<ul>';
      Object.keys(jsonData).forEach(function(key) {
        html += '<li>' + key + ': ' + JSON.stringify(jsonData[key]) + '</li>';
      });
      html += '</ul>';

      $('#output').html(html);
    },
    error: function() {
```

```
        $('#output').text('Error loading or parsing JSON file.');
      }
    });
  });
 </script>
</body>
</html>
```

Sample JSON File (data.json)

```
{
 "year": 2025,
 "technologies": ["React Native", "Expo"],
 "exp": [
  {
   "name": "Mobile App",
   "year": 2023,
   "domain": "HTML, CSS, JavaScript",
   "address": "New York",
   "skills": ["HTML", "CSS", "JavaScript"]
  }
 ]
}
```

Sample Text File (textfile.txt)

This is a sample text file for AJAX demonstration.

DETAILED EXPLANATION

---

1. Purpose of AJAX

AJAX allows data to be loaded from a server without refreshing the webpage.

✓ Faster
✓ Better user experience
✓ Partial page update

---

2. Plain AJAX (Vanilla JavaScript)

```
var xhr = new XMLHttpRequest();
```

✓ Creates an AJAX request object

```
xhr.open('GET', 'textfile.txt', true);
```

✓ GET → fetch data
✓ true → asynchronous request
xhr.onload = function() {
✓ Executes when response is received
xhr.responseText
✓ Contains text file content
📌 **Output:** Displays text file content inside #output div.

---

3. AJAX Using jQuery ($.ajax())
$.ajax({
url: 'textfile.txt',
method: 'GET',
✓ Simplified AJAX method
✓ Less code compared to plain AJAX
success: function(data)
✓ Runs if file loads successfully
error: function()
✓ Runs if file fails to load
📌 **Advantage:** Easier and cleaner syntax.

---

4. jQuery getJSON() Method
$.getJSON('data.json', function(data) {
✓ Special method to load **JSON files only**
✓ Automatically converts JSON to JavaScript object
JSON.stringify(data, null, 2)
✓ Converts object back to formatted string
✓ 2 → indentation for readability
📌 **Output:** Displays formatted JSON data.

---

5. Parsing JSON using JSON.parse()
JSON.parse(data)
✓ Converts JSON string into JavaScript object
Object.keys(jsonData)
✓ Retrieves all keys from JSON
key : value
✓ Displays JSON data in list format
📌 **Used when JSON is received as a string**

6. Why Multiple AJAX Methods Are Used

Method	Purpose
Plain AJAX	Understand core AJAX
jQuery AJAX	Simplified coding
getJSON	Load JSON easily
JSON.parse	Convert JSON to object

FINAL EXAM SUMMARY

This program demonstrates different AJAX techniques using JavaScript and jQuery to load text and JSON data dynamically without reloading the webpage.

Okay 👍 I'll tell you **clearly, step-by-step** how to run **8A (Visitor Counter)**.
Follow **exactly** in this order.

🟢 **PRE-CHECK**

Make sure in **XAMPP Control Panel**:

- **Apache → GREEN**
- **MySQL → GREEN** (MySQL not required for 8A, but no problem if running)

🔹 **STEP 1: Create the PHP file**

1. Open **Notepad**
2. Paste your **8A code**
3. Save as:

track.php

📍 Save location:

C:\xampp\htdocs\

🔹 **STEP 2: Create counter file**

1. Go to:

C:\xampp\htdocs\

2. Right-click → **New → Text Document**
3. Name it:

counter.txt

4. Open counter.txt
5. Type:

0

6. Save and close

⚠️ track.php and counter.txt **must be in the same folder**

🔹 **STEP 3: Start Apache**

1. Open **XAMPP Control Panel**
2. Click **Start** next to **Apache**
3. Apache should turn **GREEN**

🔹 **STEP 4: Run the Program**

1. Open **any browser**
2. Type:

http://localhost/track.php

3. Press **Enter**

🔹 **STEP 5: Check Output**

- Page will show:

Welcome to Our Website!

You are visitor number: 1

- Refresh the page:

2 → 3 → 4 ...

🎉 Your **8A is working correctly**

❗ **COMMON MISTAKES (DON'T DO)**

❌ Opening track.php by double-click
❌ Keeping counter.txt in another folder
❌ Apache not running

🧠 **INTERNAL EXAM ANSWER (Say this)**

"I stored the visitor count in a text file and updated it using PHP file handling functions."

Here is the **correct way to execute your 8B PHP program** step-by-step.

✅ **HOW TO EXECUTE YOUR 8B PROGRAM**

You must follow **all steps exactly**, otherwise it will not work.

STEP 1 — Put your file in the correct folder

Save your file as:

📌 **sort_students.php**

Place it inside:

C:\xampp\htdocs\students\

So final path becomes:

C:\xampp\htdocs\students\sort_students.php

STEP 2 — Start XAMPP

Open **XAMPP Control Panel**
→ Click **Start** for:

- Apache (Must be green)
- MySQL (Must be green)

STEP 3 — Create the Database

Your code uses:

$dbname = "students1";

So we create this database.

1. Open phpMyAdmin:

Open browser and go to:

http://localhost/phpmyadmin/

2. Create a new database:

Left side → Click **New**

Database name: **students1**
→ Click **Create**

STEP 4 — Create Table inside database

Now create a table named **students**

In SQL Tab paste this:

```
CREATE TABLE students (
   id INT AUTO_INCREMENT PRIMARY KEY,
   name VARCHAR(100),
   usn VARCHAR(20),
   branch VARCHAR(50),
   email VARCHAR(100),
   address VARCHAR(255)
```

);
Click **Go**

STEP 5 — Insert Sample Data

Go to **SQL** tab again and paste:

INSERT INTO students (name, usn, branch, email, address)
VALUES
('Chakrapani', '1AB21CS001', 'CSE', 'chakrapani@gmail.com', 'Bangalore'),
('Arjun', '1AB21CS002', 'CSE', 'arjun@gmail.com', 'Mysore'),
('Ravi', '1AB21CS003', 'ISE', 'ravi@gmail.com', 'Udupi'),
('Sahana', '1AB21CS004', 'ECE', 'sahana@gmail.com', 'Hassan'),
('Meghana', '1AB21CS005', 'CSE', 'meghana@gmail.com', 'Davangere');

Click **Go**

Now table is ready.

STEP 6 — Run your PHP program

Open browser and enter:

http://localhost/students/sort_students.php

If everything is correct, you will see:

✓ A table of students
✓ Sorted by **name** using Selection Sort
✓ Displayed nicely in HTML

If any error appears

Tell me the **exact error message**, I'll fix it instantly.

🔥 **If you want, I can also:**

✅ Reduce the code for LAB RECORD
or
✅ Add full explanation like 10th program
Just tell me!

Alright! Let's go **step by step** with your **8(a) PHP program**, keeping it **detailed for viva and exam**, and explaining **everything including PHP and HTML parts**.

PROGRAM 8(a)

Question

Develop a PHP program (with HTML/CSS) to keep track of the number of visitors visiting the web page and to display this count of visitors with relevant headings.

Complete Code

```
<?php
// File to store visitor count
$counterFile = "counter.txt";

// If file doesn't exist, create it and initialize with 0
if (!file_exists($counterFile)) {
    file_put_contents($counterFile, "0");
}

// Read current count from the file
$currentCount = file_get_contents($counterFile);

// Increment the count by 1
$newCount = $currentCount + 1;

// Save the new count back to the file
file_put_contents($counterFile, $newCount);
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Visitor Counter | vtucode</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            text-align: center;
            margin: 0;
            padding: 0;
            display: flex;
            flex-direction: column;
            justify-content: center;
            height: 100vh;
            background-color: #f4f4f9;
```

```
        color: #333;
    }

    .container {
        background: white;
        padding: 20px;
        box-shadow: 0 2px 10px rgba(0,0,0,0.1);
        border-radius: 8px;
        margin: 0 auto;
        width: 60%;
    }

    h1 {
        font-size: 2.5em;
        margin: 0;
    }

    p {
        font-size: 1.2em;
        color: #555;
    }

    strong {
        color: blue;
    }
  </style>
</head>
<body>
  <div class="container">
    <h1>Welcome to Our Website!</h1>
    <p>You are visitor number: <strong><?php echo $newCount; ?></strong></p>
  </div>
</body>
</html>
```

DETAILED EXPLANATION

---

1. PHP Section

a) File Handling

```
$counterFile = "counter.txt";
```

- Stores the filename that will hold the visitor count.

```
if (!file_exists($counterFile)) {
   file_put_contents($counterFile, "0");
}
```

- Checks if the file exists
- If not, creates it and initializes with 0
- Ensures program works even on first visit

---

b) Reading Current Count

```
$currentCount = file_get_contents($counterFile);
```

- Reads the current visitor count from the file
- Returns value as string (can be used as number)

---

c) Incrementing the Count

```
$newCount = $currentCount + 1;
```

- Adds 1 to the current count for this visitor

---

d) Saving Updated Count

```
file_put_contents($counterFile, $newCount);
```

- Writes the new count back into counter.txt
- Ensures the visitor number is updated for next user

---

2. HTML Section

a) <!DOCTYPE html>

- Tells browser this is **HTML5**
- Required for correct rendering

---

b) <html> and <head>

- Contains **title** and **internal CSS styling**
- Styles body and container for better appearance

---

c) Body Styling

```
body {
   font-family: Arial, sans-serif;
   background-color: #f4f4f9;
   text-align: center;
```

```
    display: flex;
    justify-content: center;
    height: 100vh;
}
```

- Font: Arial for readability
- Center content vertically and horizontally
- Light gray background for soft look

---

d) Container Styling

```
.container {
    background: white;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 2px 10px rgba(0,0,0,0.1);
    width: 60%;
    margin: 0 auto;
}
```

- White box in center
- Rounded corners (border-radius)
- Shadow for 3D effect
- Width fixed at 60% for responsive design

---

e) Displaying Visitor Count

```
<p>You are visitor number: <strong><?php echo $newCount; ?></strong></p>
```

- <?php echo $newCount; ?> → Inserts visitor count dynamically into HTML
- highlights visitor number in **blue** (via CSS)

---

3. Key Points for Viva

1. **File Handling:**
   file_exists(), file_get_contents(), file_put_contents() → used to store visitor count persistently
2. **Dynamic Content:**
   PHP executes before HTML is sent to browser → visitor number appears updated
3. **CSS Styling:**
   Enhances readability and layout of visitor count
4. **Initial Setup:**
   Program works on first visit due to file initialization

---

4. Output Example

Welcome to Our Website!
You are visitor number: **5**

Every time a user refreshes the page, the number increments by 1.

Exam One-Liner

This program uses PHP and a text file to count and display the number of visitors on a web page dynamically.

If you want, I can now **explain 8(b) (sorting student records in database using PHP)** in the **same detailed way with code + CSS + viva-proof explanation**.

Do you want me to do that next?

Perfect! Let's go **step by step** for **Lab 8(b)** with **deep, viva-proof explanation**, keeping **your code as-is**, only improving colors if needed.

PROGRAM 8(b)

Question

Develop a PHP program (with HTML/CSS) to sort student records stored in a database using Selection Sort and display them in a table.

Complete Code

(Code remains as you gave it, CSS colors are descriptive already)

```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "webtech_db";

// Connect to MySQL database
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
   die("Connection failed: " . $conn->connect_error);
}

// Fetch all student records
$sql = "SELECT * FROM students";
```

```
$result = $conn->query($sql);

$students = [];
if ($result->num_rows > 0) {
    while ($row = $result->fetch_assoc()) {
        $students[] = $row;
    }
}

// Selection Sort function to sort array by given key
function selectionSort(&$arr, $key)
{
    $n = count($arr);
    for ($i = 0; $i < $n - 1; $i++) {
        $minIndex = $i;
        for ($j = $i + 1; $j < $n; $j++) {
            if ($arr[$j][$key] < $arr[$minIndex][$key]) {
                $minIndex = $j;
            }
        }
        // Swap
        $temp = $arr[$i];
        $arr[$i] = $arr[$minIndex];
        $arr[$minIndex] = $temp;
    }
}

// Sort students by name
selectionSort($students, 'name');
?>
<!DOCTYPE html>
<head>
    <title>Sorted Student Records | vtucode</title>
    <style>
        body {
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
            background-color: #f0f2f5;
            color: #333;
```

```
    margin: 0;
    padding: 20px;
}

h2 {
    text-align: center;
    color: #4A90E2; /* Blue */
    margin-bottom: 20px;
}

table {
    width: 100%;
    border-collapse: collapse;
    background-color: #fff;
    border-radius: 10px;
    overflow: hidden;
    box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
    margin: 0 auto;
}

th, td {
    padding: 12px 15px;
    text-align: left;
    border-bottom: 1px solid #ddd;
}

th {
    background-color: #4A90E2; /* Blue */
    color: white;
    text-transform: uppercase;
    letter-spacing: 0.03em;
}

tr:hover {
    background-color: #f1f1f1; /* Light Gray on hover */
}

td {
```

```
        font-size: 0.9em;
        color: #555;
    }

    /* Responsive Table for small screens */
    @media (max-width: 768px) {
        table, th, td { display: block; width: 100%; }
        th, td { box-sizing: border-box; }
        tr { margin-bottom: 15px; display: block; box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1); }
        th { position: absolute; top: -9999px; left: -9999px; }
        td { border: none; position: relative; padding-left: 50%; text-align: right; }
        td:before {
            content: attr(data-label);
            position: absolute;
            left: 0;
            width: 50%;
            padding-left: 15px;
            font-weight: bold;
            text-align: left;
            text-transform: uppercase;
            color: #4A90E2; /* Blue label */
        }
    }
  </style>
</head>
<body>

<h2>Sorted Student Records by Name</h2>

<table>
  <thead>
    <tr>
      <th>ID</th>
      <th>Name</th>
      <th>USN</th>
      <th>Branch</th>
      <th>Email</th>
      <th>Address</th>
```

```
      </tr>
    </thead>
    <tbody>
      <?php foreach ($students as $student): ?>
        <tr>
          <td data-label="ID"><?php echo htmlspecialchars($student['id']); ?></td>
          <td data-label="Name"><?php echo htmlspecialchars($student['name']); ?></td>
          <td data-label="USN"><?php echo htmlspecialchars($student['usn']); ?></td>
          <td data-label="Branch"><?php echo htmlspecialchars($student['branch']);
?></td>
          <td data-label="Email"><?php echo htmlspecialchars($student['email']); ?></td>
          <td data-label="Address"><?php echo htmlspecialchars($student['address']);
?></td>
        </tr>
      <?php endforeach; ?>
    </tbody>
</table>

</body>
</html>
```

DETAILED EXPLANATION (Viva-Proof)

---

1. PHP Section

a) Database Connection

$conn = new mysqli($servername, $username, $password, $dbname);

- Connects PHP to MySQL database
- $servername = "localhost" → local server
- $username, $password → credentials
- $dbname = "webtech_db" → database containing students table

Viva line:

mysqli is used to connect PHP with MySQL database for fetching or manipulating data.

---

b) Fetch Records

$sql = "SELECT * FROM students";

$result = $conn->query($sql);

- SELECT * → fetch all columns from students table
- $result stores the query output

```
$students = [];
if ($result->num_rows > 0) {
   while ($row = $result->fetch_assoc()) {
      $students[] = $row;
   }
}
```

- Converts database rows into **PHP array of associative arrays**
- Each student is stored with keys: id, name, usn, branch, email, address

---

c) Selection Sort Function

function selectionSort(&$arr, $key)

- Sorts an array $arr **in-place** by column $key
- &$arr → passed by reference so original array is updated

```
for ($i = 0; $i < $n - 1; $i++) {
   $minIndex = $i;
   for ($j = $i + 1; $j < $n; $j++) {
      if ($arr[$j][$key] < $arr[$minIndex][$key]) {
         $minIndex = $j;
      }
   }
   $temp = $arr[$i];
   $arr[$i] = $arr[$minIndex];
   $arr[$minIndex] = $temp;
}
```

- Classic **Selection Sort Algorithm**:
  1. Find smallest element in unsorted part
  2. Swap with first unsorted element
  3. Repeat until fully sorted
- Sorting is done by 'name' in this program.

Viva line:

Selection Sort repeatedly finds the minimum element and swaps it to sort the array.

---

d) Sorting the Students

selectionSort($students, 'name');

- Sorts $students array by student names **alphabetically**

---

2. HTML Section

- Displays the sorted student data in a **responsive table**

- Uses **<table>, <thead>, <tbody>, <tr>, <th>, <td>**
- data-label is used for **mobile view** to show headings next to values

---

a) Table Styling

- th { background-color: #4A90E2; color: white; } → header in **blue**
- tr:hover { background-color: #f1f1f1; } → **hover effect**
- td:before { content: attr(data-label); } → shows label in small screens

---

b) Table Content

```
<?php foreach ($students as $student): ?>
  <tr>
    <td data-label="ID"><?php echo htmlspecialchars($student['id']); ?></td>
    ...
  </tr>
<?php endforeach; ?>
```

- Loops through $students array
- htmlspecialchars() → prevents HTML injection, ensures safe output
- Each row contains student details

---

c) Responsive Design

- Uses @media queries
- Converts table into **block layout** for smaller screens
- Each <td> shows **column name before value** for clarity

---

3. Key Points for Viva

1. **Database Operations:** Fetch records using mysqli
2. **Sorting Algorithm:** Selection Sort used to sort PHP array
3. **Dynamic Table:** PHP + HTML outputs sorted student records
4. **Responsive Table:** data-label ensures mobile-friendly display
5. **Security:** htmlspecialchars() prevents code injection

Exam One-Liner

This program fetches student records from MySQL, sorts them using Selection Sort in PHP, and displays them in a responsive, styled HTML table.

If you want, I can **now give Lab 9 (jQuery) and Lab 10 (AJAX) detailed explanations with code + viva-proof notes**, same way as Labs 8(a) & 8(b).

Do you want me to continue with that next?

Perfect! I'll make a **ready-to-revise viva sheet** for you. I'll take the **first 50 questions** I listed and provide **concise, exam-ready answers** (1–2 lines each) that you can **memorize or copy into a sheet**. I'll keep it **structured** and **easy to read**.

Web Technology Lab – Viva Quick Revision Sheet

HTML / CSS Questions (1–20)

1. **What is HTML?**
   HTML (HyperText Markup Language) is used to structure content on the web.
2. **Types of HTML tags?**
   Block-level (e.g., , <p>) and inline (e.g., , <a>).
3. **Difference between <head> and <body>?**
   <head> stores meta-information, <body> contains visible content.
4. **Purpose of <title>?**
   Sets page title in the browser tab.
5. **Explain <p> tag.**
   Defines a paragraph; adds vertical spacing before and after automatically.
6. **Difference between <h1> to <h6>?**
   <h1> is largest heading, <h6> is smallest.
7. **What is <marquee>?**
   Creates scrolling text; obsolete and not recommended.
8. **Purpose of <blockquote>?**
   Defines a block of quoted text; usually indented.
9. **What is <pre> used for?**
   Displays preformatted text preserving spaces and line breaks.
10. **Difference between <b> and ?**
    <b> = bold text; = important/bold with semantic meaning.
11. **Difference between <i> and ?**
    <i> = italic style; = emphasized text (semantic).
12. **Inline vs Block elements?**
    Inline (span, a) – flows with text; Block (div, p) – starts on new line.
13. **Purpose of ?**
    Used to style part of text inline without breaking line.
14. **HTML comment syntax?**
    <!-- This is a comment --> – not displayed in browser.
15. **Difference between id and class?**
    id = unique element, class = group of elements.
16. **What is a hyperlink?**
    Link – navigates to another page or resource.
17. **Types of CSS?**
    Inline, internal (<style>), external (.css file linked with <link>).

18. **What is a pseudo-class?**
    Special state of element, e.g., :hover, :first-child.
19. **Using color names instead of hex?**
    Example: background-color: red; instead of #FF0000.
20. **How does flex work in CSS?**
    display: flex creates a flexible container to align items in a row or column.

Forms / Input Questions (21–30)

21. **Purpose of <form>?**
    To collect user input and submit to server.
22. **post vs get method?**
    get = visible in URL; post = hidden and secure.
23. **Input types: text, email, password, date**
    Text = any text; Email = validates email format; Password = hides input; Date = date picker.
24. **Radio button and same name?**
    Only one option can be selected in the same name group.
25. **What is a checkbox?**
    Allows multiple selections from options.
26. **What is a textarea?**
    Multi-line input field for user text.
27. **Use of required attribute?**
    Makes field mandatory before submitting.
28. **Difference between submit and reset?**
    Submit = send form; Reset = clear all fields.
29. **How to create a dropdown list?**
    <select><option>Option1</option></select>
30. **What is form validation?**
    Ensures correct input format before submission.

Tables / Layout Questions (31–36)

31. **Difference between <thead>, <tbody>, <tfoot>?**
    <thead> = table header; <tbody> = main data; <tfoot> = footer.
32. **How do colspan and rowspan work?**
    colspan = merge columns; rowspan = merge rows.
33. **Alternate row colors in table?**
    CSS: tr:nth-child(odd){background-color: lightgray;}
34. **What is <caption>?**
    Provides table title.
35. **Difference between <section> and <article>?**
    Section = group of content; Article = independent content piece.

36. **Semantic tags <header>, <footer>, <aside>, <figure>**
    <header> = page header; <footer> = page footer; <aside> = sidebar; <figure> = media with caption.

JavaScript Questions (37–45)

37. **What is JavaScript?**
    Client-side scripting language for interactivity.
38. **Difference between var, let, const**
    var = function-scoped; let & const = block-scoped; const = constant.
39. **Get element by ID?**
    document.getElementById("id")
40. **Explain onclick event**
    Triggers a function when element is clicked.
41. **parseInt() vs parseFloat()**
    parseInt = integer; parseFloat = decimal numbers.
42. **Arrays & Objects in JS?**
    Array = list of values; Object = key-value pairs.
43. **What is JSON?**
    JavaScript Object Notation – lightweight data exchange format.
44. **Difference: JSON.parse() vs JSON.stringify()**
    Parse = JSON → JS object; Stringify = JS object → JSON string.
45. **What is a function?**
    Block of code that performs a task.

---

jQuery / AJAX Questions (46–50)

46. **What is jQuery?**
    JavaScript library for easier DOM manipulation and effects.
47. **Meaning of $(document).ready()**
    Runs code after DOM is fully loaded.
48. **append() vs html()**
    Append = adds content; html() = replaces content.
49. **What is AJAX?**
    Asynchronous JS + XML; loads data without reloading page.
50. **Plain AJAX vs jQuery AJAX**
    Plain = XMLHttpRequest; jQuery = simpler syntax with $.ajax().

✅ This is **ready-to-revise** and covers **all your lab programs**.

I can also make a **Part-2 sheet** for **PHP, MySQL, Node.js, Crypto, JSON parsing**, since you did Labs 8–10 too.

Do you want me to make that Part-2 sheet next?