Question 1: Creating a Webpage Layout

- **Design a webpage layout using HTML and CSS Grid:**
  - The layout should include:
    - A header at the top.
    - A sidebar on the left.
    - Main content in the center.
    - A footer at the bottom.
    - Each section must have some content.

  *Use the <header>, <aside>, <main>, <footer>, <div>

  - Use CSS Grid to divide the page into these sections.

  *Use the grid-template-areas property to define grid regions.

  display: grid; grid-template-areas; grid-template-rows; grid-template-columns;

- **Enhance the layout by adding the following:**
  - The header and footer should have a solid background color and fixed height.
  - The sidebar should have a fixed width and a scrollable area for additional content.
  - The main content should take up the remaining space.

  *CSS properties like background-color, padding, and box-shadow.

  background-color; padding; border; text-align

Question 2: Create a responsive navigation bar using HTML and CSS

- **The navigation bar should include four links: Home, About, Services, Contact.**
- **Links should be horizontally aligned.**

  Use the <nav> tag to define the navigation bar.

  Create a list of links using <ul> for an unordered list and <li> for each list item.

  Wrap the text in <a> tags to make them clickable.

Use **Flexbox** to style the <ul> container.

Set display: flex; on the <ul> and use justify-content: space-evenly; to distribute the links evenly.

- **On hover, change the background color of the links.**

  Use the :hover pseudo-class to apply styles when a link is hovered.

- **Make the navigation bar fixed at the top of the page.**

  Use position: fixed; and top: 0; to position the bar at the top.

  Set width: 100%; to ensure it spans the full width of the page.

- **Add a background gradient.**

  Use the background: linear-gradient(); property to create the gradient.

- **Use Flexbox to distribute the links evenly.**

Use media queries (@media) to detect screen size and apply different styles for smaller screens.

Change flex-direction to column on smaller screens.

Question 3: Create a Pricing Table

- Include **three pricing tiers**: **Basic**, **Standard**, and **Premium**.

Each tier must include:

- A **heading** for the plan name using a <h2> tag.
- A **price** using a <p> tag or <div>.
- A **list of features** using an unordered list (<ul> and <li>).
- A **button** to select the plan using a <button> tag.

Enhance the Table with the Following CSS Effects:

- **Borders and Shadows:**
  Use border to define a border around each pricing card.
  Use box-shadow for a "lifted" effect.
- **Hover Effects:**
  Apply :hover on the feature list items to display additional details (using title or tooltips).
  Scale the button and change its color on hover (transform and background-color).
- **Highlighting:**

Use a distinct border or background color for the "Standard" plan to indicate it is the most popular.

- **Ribbon Effect:**
  Add a "Popular" ribbon on the top-left corner of the "Standard" plan using a pseudo-element (::before or ::after).

Question 4: Design and implement a calendar using HTML and CSS

- **Basic Structure**:
  - Write the HTML code to create a calendar for a specific month (e.g., January 2025).
  - The calendar should include:
    - A heading with the month and year (e.g., "January 2025").
    - A row displaying the days of the week (e.g., Sunday, Monday, etc.).
    - Rows for the dates, arranged correctly to match the days of the week.

Tags to be used: <header>, <div>, <h1>, <div>, <span>

- **Styling with CSS**:
- Style the calendar so that:
  - Each cell (day) has a border, and the entire calendar is centered on the page.
  - Weekends (Saturday and Sunday) have a different background color.
  - The current date is highlighted with a special style.
  - Ensure proper spacing and alignment of text within the calendar cells.

Use: border, text-align, margin, padding, background-color

- **Enhanced Features**:
  - Display tooltips on each day, showing a message like "Click to add an event."
  - Use CSS Grid or Flexbox to create the layout of the calendar.

Use: display: grid; grid-template-columns; flexbox; :hover; content: attr(title)

- **Responsive Design**:
- Modify the calendar to be responsive:

  - On smaller screens, stack the days vertically for better readability.
  - Ensure the calendar resizes gracefully without losing structure.

Use: @media, grid-template-columns: 1fr

- **Additional Challenges (Optional)**:
- Add event placeholders for specific dates (e.g., "Meeting at 3 PM" on January 15).

Use: title, <p>, <span>, ::after