1. **Create a Simple Webpage:** Create a webpage with a title “My First Webpage” that includes a header, paragraph, and an image. The image should be centered, and the header should be in bold.
2. **Hyperlink Creation:** Create a navigation menu with three links: “Home”, “About Us”, and “Contact Us”. The “Home” link should open a new page, the “About Us” link should scroll to a section within the same page, and the “Contact Us” link should open the mail client.
3. **Form Design:** Create a form with the following fields:
   * Name (Text Input)
   * Email (Email Input)
   * Password (Password Input)
   * Gender (Radio Buttons)
   * Hobbies (Checkboxes)
   * Submit Button

* Validate the email and password fields.

1. **Table Creation:** Create a table with 3 rows and 3 columns. The first row should be the header (containing “Name”, “Age”, and “City”). The second and third rows should contain data for two different individuals.
2. **Ordered and Unordered Lists:** Create two lists on a webpage:
   * An ordered list of your top 5 favorite movies.
   * An unordered list of your favorite hobbies.
3. **Image Gallery:** Create a simple image gallery with three images arranged in a row. Each image should have a caption beneath it.
4. **Video and Audio Embedding:** Embed a video and an audio file on the webpage. The video should have controls for play, pause, and volume. The audio file should auto-play on page load.
5. **Collapsible Sections:** Using <details> and <summary>, create three collapsible sections labeled “Introduction”, “Skills”, and “Projects”. Each section should contain appropriate content related to the label.
6. **Iframe Integration:** Embed a Google Map of your city’s location on the webpage using an iframe. Ensure that the iframe has a width of 600 pixels and height of 400 pixels.
7. **HTML5 Semantic Tags:** Create a webpage using HTML5 semantic tags (<header>, <nav>, <section>, <article>, <footer>). Design a basic blog page layout using these tags with a navigation bar, main content, and a footer containing contact information.

Here are the first 5 Git questions and questions 19 and 20:

1. **Git Initialization:** Create a new directory named “my\_project” and initialize it as a Git repository. Add a file called index.html, stage it, and commit it with the message “Initial commit.”
2. **Cloning a Repository:** Clone a public repository from GitHub (you can provide a sample repository URL). After cloning, display the list of branches in the repository using Git commands.
3. **Branch Creation and Checkout:** In your Git repository, create a new branch called feature-login. Switch to this branch, create a file login.html, and commit the file to the feature-login branch with a relevant commit message.
4. **Merging Branches:** Merge the changes from the feature-login branch into the main branch. Before merging, ensure you are on the main branch. Handle any potential merge conflicts, and commit the merge.
5. **Viewing Commit History:** Display the commit history of the current branch, showing only the commit messages and hash IDs in a simplified format. Then, display the commit history graphically using Git commands.
6. **Git Pull and Push:** Demonstrate how to pull the latest changes from a remote repository into your local repository. Make some changes locally, commit them, and then push the changes to the remote repository.
7. **Resolving Conflicts:** Simulate a merge conflict situation by modifying the same line in a file on two different branches. Show the process of resolving the conflict and committing the resolution.

### 18. **Box Model: Padding and Margin**

**HTML:**

<div class="box">  
 Content inside the box  
</div>

**Question:** Write the CSS to create a box with the following: - A width of 300px and a height of 200px. - Add 20px padding inside the box. - Add a 10px margin outside the box. - Apply a solid border of 2px in blue.

### 19. **Box Model: Border and Box-Sizing**

**HTML:**

<div class="container">  
 <div class="box-1">Box 1</div>  
 <div class="box-2">Box 2</div>  
</div>

**Question:** Write the CSS to: - Set the width of both boxes to 150px and height to 100px. - Add 5px solid black borders to both boxes. - Ensure that padding does not increase the size of the boxes (use box-sizing).

### 20. **Units: Relative and Absolute Units**

**HTML:**

<div class="text-container">  
 <p class="text">This is some text.</p>  
</div>

**Question:** Write the CSS to: - Set the font size of the text to 2em. - Set the width of .text-container to 50% of the viewport width. - Add 1rem padding to .text-container.

### 21. **Units: Viewport Units**

**HTML:**

<div class="header">  
 <h1>Welcome to My Website</h1>  
</div>

**Question:** Write the CSS to: - Set the height of the header to 50vh. - Add padding of 5vw to the header. - Center the text horizontally and vertically inside the header.

### 22. **Position: Fixed Header**

**HTML:**

<div class="header">Fixed Header</div>  
<div class="content">  
 <p>Some long content goes here...</p>  
</div>

**Question:** Write the CSS to: - Make the .header fixed at the top of the page with a width of 100%. - Give it a height of 60px, a background color, and make the content scroll underneath it. - Add some padding inside the .content so that it doesn’t overlap with the header.

### 23. **Position: Absolute and Relative**

**HTML:**

<div class="container">  
 <div class="relative-box">Relative Box</div>  
 <div class="absolute-box">Absolute Box</div>  
</div>

**Question:** Write the CSS to: - Make the .container 400px by 400px with a border. - Position .relative-box relatively with a top offset of 20px and left offset of 20px. - Position .absolute-box absolutely at the bottom-right corner of .container.

### 24. **Flexbox: Centering with Flex**

**HTML:**

<div class="flex-container">  
 <div class="box">Box 1</div>  
 <div class="box">Box 2</div>  
 <div class="box">Box 3</div>  
</div>

**Question:** Write the CSS to: - Apply Flexbox to .flex-container so that the .box elements are centered horizontally and vertically in the container. - The .flex-container should have a height of 300px. - Each .box should have a width of 100px and height of 100px.

### 25. **Flexbox: Space Between Items**

**HTML:**

<div class="flex-container">  
 <div class="item">Item 1</div>  
 <div class="item">Item 2</div>  
 <div class="item">Item 3</div>  
</div>

**Question:** Write the CSS to: - Make the .flex-container a Flexbox container. - Space the .item elements evenly using justify-content: space-between. - Add a border around each .item and set the height of .flex-container to 200px.

### 26. **Flexbox: Column Layout**

**HTML:**

<div class="flex-container">  
 <div class="box">Box 1</div>  
 <div class="box">Box 2</div>  
 <div class="box">Box 3</div>  
</div>

**Question:** Write the CSS to: - Set the .flex-container to display its .box elements in a vertical column. - Center the items horizontally within the container. - Give each .box a height of 80px and a different background color.

### 27. **Flexbox: Wrapping Items**

**HTML:**

<div class="flex-container">  
 <div class="item">Item 1</div>  
 <div class="item">Item 2</div>  
 <div class="item">Item 3</div>  
 <div class="item">Item 4</div>  
 <div class="item">Item 5</div>  
</div>

**Question:** Write the CSS to: - Make the .flex-container a Flexbox container that allows wrapping. - Ensure that the .item elements wrap onto the next line when the container is too narrow. - Set the width of each .item to 150px and the height to 100px.