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.

<header><b>Document</b></header>

<p>Some content</p>

<center><img src=“” alt=“”></center>

1. **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.

<nav>

<a href=“link” target=“\_blank”>Home</a>

<a href=“link” target=“-self”>About us</a>

<a href=“<mailto:abc@gmail.com>>Contact us</a>

</nav>

1. **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.

<label for="name">Name:</label>

<input type="text" id="name" name="name"><br><br>

<label for="email">Email:</label>

<input type="email" id="email" name="email" required><br><br>

<label for="password">Password:</label>

<input type="password" id="password" name="password" required><br><br>

<label>Gender:</label>

<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><br><br>

<label>Hobbies:</label>

<input type="checkbox" id="hobby1" name="hobbies" value="Reading">

<label for="hobby1">Reading</label>

<input type="checkbox" id="hobby2" name="hobbies" value="Traveling">

<label for="hobby2">Traveling</label>

<input type="checkbox" id="hobby3" name="hobbies" value="Sports">

<label for="hobby3">Sports</label><br><br>

<button type="submit">Submit</button>

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.

<table>

<tr>

<th>Name</th>

<th>Age</th>

<th>City</th>

</tr>

<tr>

<td>Name1</td>

<td>Age1</td>

<td>City1</td>

</tr>

<tr>

<td>Name2</td>

<td>Age2</td>

<td>City2</td>

</tr>

</table>

1. **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.

<list>

<ol>

<li>Movie1</li>

<li>Movie2</li>

<li>Movie3</li>

<li>Movie4</li>

<li>Movie5</li>

</ol>

<ul>

<li>Hobby1</li>

<li>Hobby2</li>

<li>Hobby3</li>

<li>Hobby4</li>

<li>Hobby5</li>

</ul>

</list>

1. **Image Gallery:** Create a simple image gallery with three images arranged in a row. Each image should have a caption beneath it.

<table>

<tr>

<td>

<img src=“link” alt=“”>

<p>Some content</p>

</td>

<td>

<img src=“link” alt=“”>

<p>Some content</p>

</td>

<td>

<img src=“link” alt=“”>

<p>Some content</p>

</td>

</tr>

</table>

1. **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.

<video controls width="600">

<source src="video-file.mp4" type="video/mp4">

</video>

<audio autoplay>

<source src="audio-file.mp3" type="audio/mp3">

</audio>

1. **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.

<details>

<summary>Introduction</summary>

<p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Quidem distinctio dolorem quod at eius blanditiis temporibus soluta aperiam officiis, enim non assumenda tempore alias nobis illum vero eligendi totam beatae?</p>

</details>

<details>

<summary>Skills</summary>

<p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Cupiditate corrupti dolor ad quisquam, officiis aliquid enim reiciendis, modi, quaerat sint consequatur corporis libero a. Delectus aut ratione esse quos amet..</p>

</details>

<details>

<summary>Projects</summary>

<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Labore nostrum corporis</p>

</details>

1. **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.

<iframe src="https://www.google.com/maps/embed?pb=!1m18!1m12!1m3!1d15224.172688493027!2d78.39559795!3d17.457647650000002!2m3!1f0!2f0!3f0!3m2!1i1024!2i768!4f13.1!3m3!1m2!1s0x3bcb91f08e54eb3f%3A0x3128d84171b9c1a0!2sTaaza%20Kitchen!5e0!3m2!1sen!2sin!4v1729264840111!5m2!1sen!2sin" width="600" height="450" style="border:0;" allowfullscreen="" loading="lazy" referrerpolicy="no-referrer-when-downgrade" frameborder="0">

1. **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.

<header>

<section>

<article>

<h2>Blog Post Title</h2>

<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus lacinia odio vitae vestibulum vestibulum.</p>

<p>Another paragraph of the blog post content goes here...</p>

</article>

<aside>

<h3>About Me</h3>

<p>Short bio or other relevant information.</p>

</aside>

</section>

<footer>

<p>Contact us at: email@example.com</p>

<p>&copy; 2024 My Blog</p>

</footer>

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.”

Create my\_project directory

git init

git add index.html

git commit-m “Initial commit”

1. **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.

git clone url

git fetch

git branch-a

1. **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.

git checkout -b feature-login

Create login.html file

git add .

1. **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.

git switch main

git pull origin main

git merge feature-login

If there are merge conflicts, resolve them manually

git add .

git commit-m “Merge feature-login to main”

1. **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.

git log all

git log —graph

1. **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.

git pull origin main

git fetch

Make changes to html file

git add .

git commit-m “Main”

git push origin main

1. **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.

git checkout-b feature-update-login

Modify line in html file

git add .

git commit -m “main”

git switch main

Modify same line in html file

git add .

git commit -m “Merge”

git merge feature-update-login

Open html file to solve conflict

git add .

### 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.

.box{

width:300px;

height:200px;

padding:20px;

margin:10px;

border:2px solid 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).

.container{

margin:0;

padding:0;

box-sizing:border-box;

}

.box-1{

height:100px;

width:150px;

border:5px solid;

padding:10px;

}

.box-2{

height:100px;

width:150px;

border:5px solid;

padding:10px;

}

### 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.

.text{

font-size:2em;

}

.text-container{

width:50vw;

padding:1rem;

}

### 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.

.header{

height:50vh;

padding:5vw;

text-align:center

justify-content:center;

}

### 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.

.header{

width:100%;

height:60px;

position:fixed;

background-color:burlywood;

}

.content{

padding:50px;

}

### 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.

.container{

width:400px;

height:400px;

border:2px solid;

position:relative

}

.relative-box{

top:20px;

left:20px;

position:relative;

}

.absolute-box{

position:absolute;

bottom:0px;

right:10px;

}

### 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.

.flex-container{

height:300px;

width:

text-align:center;

Justify-content:center;

}

.box{

height:100px;

width: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.

.flex-container{

display:flex;

justify-content:space-between;

height:200px;

}

.item{

border: 2px solid;

}

### 26. Flexbox: Column Layout

**HTML:**

<div class="flex-container">  
 <div class="box1">Box 1</div>  
 <div class="box2">Box 2</div>  
 <div class="box3">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.

.flex-container{

display:flex;

flex-direction:column;

align-items:center;

}

.box1 {

height:80px;

background-color:pink;

}

.box2{

height:80px;

background-color:gray;

}

.box3 {

height:80px;

background-color:purple;

}

### 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.

.flex-container{

display:flex;

flex-wrap:wrap;

gap:10px;

}

.item{

width:150px;

height:100px;

border:2px solid;

}