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.  
   **Answer:**  
   <head>

    <title>My First Webpage</title>

</head>

<body>

    <header>

        This is header content

    </header>

    <p>

        This is paragraph content

    </p>

   <center><img src="../Day3/civic type r.jpg" alt="image"></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.  
   **Answer:**  
   <nav>

<span><a href="./home.html" target="\_blank">Home</a></span>

<span><a href="#aboutUs">About Us</a></span>

<span><a href="mailto:example@email.com">Contact Us</a></span>

</nav>

<p id="aboutUs">This is some data</p>

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.  
  **Answer:**  
  <form action="./success.html">
* <input type="text" id="Name" name="Name">
* <input type="email" name="email" id="email">
* <input type="text" name="password" id="password" pattern="[a-zA-Z0-9]{8-16}!$%">
* <input type="radio" name="gender" id="male" value="male">
* <label for="male">Male</label>
* <input type="radio" name="gender" id="female" value="female">
* <label for="female">Female</label>
* <input type="checkbox" name="hobbies" id="singing" value="singing">
* <label for="singing">Singing</label>
* <input type="checkbox" name="hobbies" id="dancing" value="dancing">
* <label for="dancing">dancing</label>
* <input type="submit" value="submit">
* </form>

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.  
   **Answer:**  
   <table>

        <th>

            <td>Name</td>

            <td>Age</td>

            <td>City</td>

        </th>

        <tr>

            <td>Maruthi</td>

            <td>21</td>

            <td>Hyderabad</td>

        </tr>

        <tr>

            <td>Srikanth</td>

            <td>21</td>

            <td>Chennai</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.

**Answer:**

<ol>

<li>Pokiri</li>

<li>Bahubali</li>

<li>Bahubali 2</li>

<li>Ala Vaikantapuram lo</li>

<li>S/o Satyamurthy</li>

</ol>

<ul>

<li>Video Games</li>

<li>Listening to music</li>

<li>Bike riding</li>

<li>Long walks</li>

<li>Exploring new places</li>

</ul>

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

<img src="./image1.png" alt="image1">

<figcaption>First Image</figcaption>

</figure>

<figure>

<img src="./image2.png" alt="image2">

<figcaption>Second Image</figcaption>

</figure>

<figure>

<img src="./image3.png" alt="image3">

<figcaption>Third Image</figcaption>

</figure>

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.  
   **Answer:**  
   <video src="./video.mp4" controls></video>

<audio src="./song.mp3" autoplay></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.  
   **Answer:**  
   <details>

<summary>Introduction</summary>

<p>Hello, My Name is Maruthi</p>

</details>

<details>

<summary>Skills</summary>

<p>HTML, CSS, Java</p>

</details>

<details>

<summary>Projects</summary>

<p>My School project details</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.  
   **Answer:**  
   <div class="container">

<iframe src="https://www.google.com/maps/DKF" frameborder="0" width="600px" height="400px"></iframe>

</div>

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.  
   **Answer:** <!-- header -->  
   <header>

<!-- nav bar -->

<nav>Welcome to McDonald's</nav>

</header>

<!-- section -->

<section>

<!-- article -->

<article>

<p>This is content about McDonald's</p>

</article>

</section>

<!-- footer -->

<footer>

Contact Us

</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.”  
   **Answer:** sudo mkdir my\_project  
   cd mkdir my\_project  
   git init  
   sudo vi index.html  
   git add .   
   git commit -m “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.  
   **Answer:**  
   git clone <https://github.com/sample>  
   git branch --list
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.  
   **Answer:**   
   git checkout -b feature-login  
   git switch feature-login  
   sudo vi index.html  
   git add .   
   git commit -m “new file index.html”
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.  
   **Answer:**  
   git switch main  
   git merge feature-login  
   git add .  
   git commit -m “resolved merge conflicts”
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.  
   **Answer:**  
   git log   
   git log --graph
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.  
   **Answer:**  
   git pull origin main  
   sudo cat index.html   
   git add .   
   git commit -m “pulled changes and modified file”
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.  
   **Answer:**  
   git switch feature  
   sudo cat index.html  
   git switch master  
   sudo cat index.html  
   git merge feature  
   git status  
   git add .  
   git commit -m “resolved merge conflicts”

### 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.  
**Answer:**  
.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).  
**Answer:**  
.box-1, .box-2 {

width: 150px;

height: 150px;

border: 5px solid black;

box-sizing: border-box;

}

### 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.  
**Answer:**  
.text-container {

padding: 1rem;

}

.text {

font-size: 2em;

width: 50vw;

}

### 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.  
**Answer:**  
.header {

height: 50vh;

padding: 5vw;

text-align: center;

vertical-align: middle ;

}

### 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.  
**Answer:**  
.header {

position: fixed;

width: 100%;

top: 0px;

z-index: 1;

height: 60px;

background-color: aliceblue;

}

### 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.  
**Answer:**  
.container {

width: 400px;

height: 400px;

border: 2px solid;

position: relative;

}

.relative-box {

position: relative;

top: 20px;

left: 20px;

}

.absolute-box {

position: absolute;

bottom: 0px;

right: 0px;

}

### 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.  
**Answer:**  
.flex-container {

display: flex;

justify-content: center;

align-items: center;

height: 300px;

}

.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.  
**Answer:**   
.flex-container {

display: flex;

justify-content: center;

height: 200px;

}

.item {

border: 2px solid;

}

### 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.  
**Answer:**

.flex-container {

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

}

.box1, .box2, .box3 {

height: 80px;

}

.box1 {

background-color: aliceblue;

}

.box2 {

background-color: aqua;

}

.box3 {

background-color: gray;

}

### 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.  
**Answer:** .flex-container {

display: flex;

flex-wrap: wrap;

}

.item {

width: 150px;

height: 100px;

}