TEST.

SECTION 1(3 QUNS).

1a) The role of **HTML** would be providing the structure of a web page like defining the headings and the paragraphs.

The role of **CSS** would be making the page look nice by adding fonts, colors and layouts.

JavaScript adds interactivity and logic, enabling dynamic actions such as form validation.

b) When a browser loads a page, it reads the HTML first, then applies the CSS for style, and finally runs JavaScript to make it interactive.

2. **Discussing the main factors that cause cross-browser inconsistencies and propose two practical methods to minimize them:**

Different browsers may show a page differently because they **use different engines**, or **have varying default styles and different web standards.**

The two practical methods would be:

Use standard, well-tested HTML and CSS code

Test their websites on multiple browsers to fix display issues.

3. Responsive design allows a website to adjust automatically to any screen size like phones, tablets, or computers.
Two techniques are:

1. Using **CSS media queries** to change layout based on screen width.
2. Using **flexible grids and images** that resize automatically to fit the screen.

SECTION 2(3QUNS).

The difference between semantic and non- semantic HTML.

Semantic uses tags like <header>, <main> while non-semantic uses <div> or <span>

Why semantic structure is crucial

It helps search engines understand the page better (good for SEO).

It helps screen readers and improves accessibility.

Examples

<header>My Portfolio</header>

<article>This is a blog post .</article>

Number 2

Nesting and Hierarchy in HTML
Nesting means putting HTML tags inside each other in the correct order.
Proper nesting helps browsers read and display the page correctly. Bad nesting can break the layout.

Number 3

Key Attributes

action: tells where the form data goes.

method: tells how data is sent (GET or POST).

name: gives the input field an identity.

required: makes sure the user fills in the field.

Section 3(4quns)

1a) Define each component (content, padding, border, margin).

Content: The stuff inside the box (text, images)

Padding: Space between content and border.

Border: The line around the box.

Margin: Space outside the box.

b) Imagine you have a div with a width of 200px. You add padding: 20px and border: 5px. If you forget that padding and border **add to the total width**, the div will actually take up **250px** (200 + 20×2 + 5×2).

If this div is next to another div in a row, it can **break the layout** by causing the second div to drop below the first or create unwanted horizontal scrollbars.

2) a) Discuss the key differences between Flexbox and Grid.
b) Explain when each is more appropriate to use.

Flexbox: Works in one direction (row or column), good for menus or small layouts.

Grid: Works in rows and columns, good for big layouts like pages or galleries.

Use Flexbox: When you just want to align items in a line.
Use Grid: When you want full control of rows and columns.

3) **a) How CSS contributes to separation of concerns:**

- CSS keeps the **design and style separate from the HTML content**.
- This means the HTML handles **structure and content**, while CSS handles **colors, layout, fonts, etc.**
- It makes the code cleaner and easier to read.

b) How violating it affects collaboration and scalability:

- **Using inline styles** mixes style with content.
- Harder for teams: developers and designers can't work separately.
- Harder to update: if you want to change a style across the site, you have to edit each element individually.
- Can cause **inconsistent design** and slower development on big projects.

4) Examine the impact of CSS specificity on debugging and maintenance.
Provide an example of conflicting CSS rules and explain which one the browser applies and why.

Impact of CSS Specificity on Debugging and Maintenance:

- CSS specificity determines which style the browser applies when multiple rules target the same element.
- Higher specificity rules **override** lower ones.
- Misunderstanding specificity can make it **hard to figure out why a style isn't working**, which slows down debugging and maintenance.

```
p { color: blue; }       /* Less specific */

#main p { color: red; }  /* More specific */
```