Assignment –2

Question 1: (5 Marks)

Given an array and a chunk size, return a chunked array. A chunked array contains the original elements in arr but consists of subarrays each of length size. The length of the last subarray may be less than size if arr.length is not evenly divisible by size.
Please solve it without using lodash's _.chunk function.

Example 1:
Input: arr = [1,2,3,4,5], size = 1
Output: [[1],[2],[3],[4],[5]]
Explanation: The arr has been split into subarrays each with 1 element.

Example 2:
Input: arr = [1,9,6,3,2], size = 3
Output: [[1, 9, 6], [3, 2]]
Explanation: The arr has been split into subarrays with 3 elements. However, only two elements are left for the 2nd subarray.

Example 3:
Input: arr = [8, 5, 3, 2, 6], size = 6
Output: [[8, 5, 3, 2, 6]]
Explanation: Size is greater than arr.length; thus, all elements are in the first subarray.

Example 4:
Input: arr = [], size = 1
Output: []
Explanation: There are no elements to be chunked so an empty array is returned.

Constraints:

- arr is a string representing the array.
- 2 <= arr.length <= 105
- 1 <= size <= arr.length + 1

Question 2: Dynamic Theme Architect (10 Marks)

Task

Create a webpage with three buttons:

- Morning
- Afternoon
- Night

When a button is clicked, apply a predefined theme object that dynamically updates the page styling.

Requirements

Each Theme Object must define:

- backgroundColor
- textColor
- headingStyle

On button click:

- Change page background color using **DOM manipulation**
- Change text color using **DOM manipulation**
- Modify all <h1> elements using querySelectorAll()
- Apply heading styles dynamically through JavaScript
- No page reload allowed

Students must:

- Use event listeners
- Use object-based theme design
- Apply all styling changes via JavaScript (not CSS classes only)

Students should use a minimal structure similar to the following to demonstrate functionality:

```
<!DOCTYPE html>
<html>
<head>
  <title>Dynamic Theme Architect</title>
</head>
<body>


  <h1>Welcome to Theme Architect</h1>
  <h1>Dynamic Styling Demo</h1>


  <button id="morningBtn">Morning</button>
  <button id="afternoonBtn">Afternoon</button>
  <button id="nightBtn">Night</button>


  <script src="script.js"></script>
</body>
</html>
```

All functionality must be implemented inside script.js

Question3: Grocery Store Inventory Dashboard (20 Marks)

Scenario

You are developing a dashboard for a grocery store manager. You are provided with a data source (array of objects) and a blank container (#inventory-grid) in your HTML.

Data Source

```
const inventory = [
  { id: 1, name: "Organic Bananas", category: "Fruit", stock: 12, price: 0.99 },
  { id: 2, name: "Whole Milk", category: "Dairy", stock: 2, price: 3.50 },
  { id: 3, name: "Sourdough Bread", category: "Bakery", stock: 0, price: 4.25 },
  { id: 4, name: "Roasted Almonds", category: "Snacks", stock: 25, price: 8.00 }
];
```

Requirements

Render Engine (8 Marks)

Write a function renderInventory(data) that:

- Loops through the array
- Dynamically creates a product card for each item
- Displays name, category, price, and stock
- Appends all cards inside #inventory-grid

Mandatory:

Use document.createElement() and appendChild() for DOM construction (do not rely entirely on innerHTML).

Conditional Visual Logic (4 Marks)

- If stock === 0
  - Add CSS class .out-of-stock
  - Display “Restock Needed” badge
- If stock < 5 (but not 0)
  - Display stock count in orange

Use proper DOM manipulation (classList, style, etc.).

Tax Toggle (4 Marks)

Add a button labeled **“Show Price with Tax”**.

- On click, update prices to include 10% tax (price × 1.1)
- Re-render or dynamically update the DOM
- No page reload allowed

Quick Delete (4 Marks)

Each card must include a **Remove** button.

- On click, remove the card from the DOM
- Remove the corresponding item from the inventory array

Question 4: Responsive Layout Design Using CSS (20 Marks)

Scenario

You are required to design a responsive product layout using three different CSS layout techniques:

- **Float**
- **Flexbox**
- **CSS Grid**

A basic HTML structure is provided below.

Given HTML (Use This Only)

```
<!DOCTYPE html>
<html>
<head>
    <title>Responsive Layout Task</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>

    <h1>Product Dashboard</h1>

    <div class="container">
        <div class="card">Product 1</div>
        <div class="card">Product 2</div>
        <div class="card">Product 3</div>
        <div class="card">Product 4</div>
    </div>

</body>
</html>
```

Requirements

Part A: Layout Implementation (15 Marks)

Using the same HTML structure:

1. Create three separate CSS implementations:
   - One using **Float**
   - One using **Flexbox**
   - One using **CSS Grid**
2. The layout must:
   - Display 4 cards in a row on desktop
   - Display 2 cards per row on tablet
   - Display 1 card per row on mobile
   - Be fully responsive using media queries
3. Basic styling required:
   - Card padding
   - Border
   - Proper spacing
   - Centered heading

Part B: Conceptual Question (5 Marks)

Write a short comparison explaining:

- The difference between **Float**, **Flexbox**, and **Grid**
- When each approach is most suitable
- Which one is best for modern responsive design and why