PYTHON DEVELOPER

TASK - 7

47. **Count Inversions**
    Objective: Count the number of inversions in an array, where an inversion is when a[i]>a[j]a[i] > a[j]a[i]>a[j] for i<ji < ji<j.
    Input: A list of integers.
    Output: The count of inversions.
    Hint: Use a modified merge sort to count inversions during the merge step.

---

48. **Find the Longest Palindromic Substring**
    Objective: Find the longest palindromic substring in a given string.
    Input: A string.
    Output: The longest palindromic substring.
    Hint: Use dynamic programming or expand around center approach.

---

49. **Traveling Salesman Problem (TSP)**
    Objective: Find the shortest possible route that visits each city once and returns to the origin city.
    Input: A list of cities and the distances between each pair of cities.
    Output: The shortest possible route and its total distance.
    Hint: Use dynamic programming or branch-and-bound for an approximate solution.

---

50. **Graph Cycle Detection**
    Objective: Detect whether a graph contains a cycle.
    Input: An undirected graph represented as an adjacency list.
    Output: `True` if the graph contains a cycle, otherwise `False`.
    Hint: Use depth-first search (DFS) with a recursion stack to detect cycles.

---

51. **Longest Substring Without Repeating Characters**
    Objective: Given a string, find the length of the longest substring without repeating characters.
    Input: A string.

Main Flow Services and Technologies Pvt. Ltd.
Contact Us. +91 9389641586, +91 97736 99074
Email-Add. contact.mainflow@gmail.com
www.mainflow.in

Output: The length of the longest substring without repeating characters.
Hint: Use the sliding window technique and a hash set or dictionary to track characters.

---

52. **Find All Valid Parentheses Combinations**
    Objective: Generate all possible valid combinations of parentheses.
    Input: An integer nnn, representing the number of pairs of parentheses.
    Output: A list of valid parentheses combinations.
    Hint: Use recursion and backtracking to generate valid combinations.

---

53. **Zigzag Level Order Traversal of Binary Tree**
    Objective: Traverse a binary tree in a zigzag level order.
    Input: A binary tree root.
    Output: A list of lists, where each list represents a level in zigzag order.
    Hint: Use two stacks to alternate between left-to-right and right-to-left traversal.

---

54. **Palindrome Partitioning**
    Objective: Partition a string such that every substring is a palindrome.
    Input: A string.
    Output: A list of lists of palindromic partitions.
    Hint: Use backtracking to explore all possible partitions.

---

7. Personal Budget Advisor

- **Description:** Build a program to track expenses and income, analyze spending patterns, and provide suggestions for saving money.
- **Challenges:**
  - Create a rule-based suggestion system for budgeting.
  - Summarize data using percentages and trends.
  - Implement error handling for incorrect user inputs.
- **Skills:** Conditional logic, file I/O, and mathematical calculations.

7. Personal Budget Advisor

Main Flow Services and Technologies Pvt. Ltd.
Contact Us. +91 9389641586, +91 97736 99074
Email-Add. contact.mainflow@gmail.com
www.mainflow.in

- **Restriction**: **No pre-built financial or statistical libraries** (e.g., `pandas`, `numpy`).
- **Reason**: This forces students to implement their own methods for handling and analyzing financial data. While tools like `pandas` can make things easier, the goal is for students to learn how to manipulate data manually using basic structures like **lists**, **dictionaries**, and **loops**. By doing so, they will gain a deeper understanding of data analysis without relying on external packages.
- **Learning Outcome**: Students will develop skills in **data manipulation**, basic **statistical analysis**, and **budgeting algorithms**, learning how to create effective **financial tracking tools** without the aid of external libraries.

Deadline Compliance

- **Restriction**: **Submit the project within 7 days** from the start date.
- **Reason**: Meeting deadlines is crucial in the real-world software development environment. This restriction helps students practice **time management** and **task prioritization**. In professional settings, tight deadlines are often the norm, and learning to meet them without compromising quality is an essential skill.
- **Learning Outcome**: Students will learn to manage their time effectively, complete projects under pressure, and **deliver results on time**, which are all important skills in the workplace.