Lab Experiment: 10 **Batch: 1 & 2**
Subject: Data Structures Lab **MCA**
Semester: 1st

Objective:

- To understand and implement AVL Trees with self-balancing properties.
- To apply heap sort for efficient sorting.
- To implement a priority queue using a heap structure.

Instructions:

Implement the following tasks in C. Use appropriate data structures and algorithms to create and manipulate AVL Trees, perform heap sorting, and demonstrate a priority queue.

Assignment 1st: AVL Tree Implementation

Definition: An AVL Tree is a self-balancing binary search tree. For any node in the tree, the height difference between its left and right subtrees is at most one.

Tasks:

- Implement insertion in an AVL tree. Ensure that after each insertion, the tree remains balanced using rotations (left rotation, right rotation, left-right rotation, right-left rotation).
- Implement deletion in an AVL tree with the necessary rebalancing steps.

Testing: Insert and delete a series of values, displaying the tree structure after each operation.

Assignment 2nd: Heap Sort Implementation

Definition: Heap sort is a comparison-based sorting algorithm that uses a binary heap (typically a max-heap).

Tasks:

- Build a max heap from an array of unsorted elements.
- Implement heap sort by repeatedly removing the root element (maximum value) and re-heapifying the tree.

Testing: Demonstrate heap sort with an example array, showing each step and the final sorted output.

Assignment 3rd: Priority Queue Using Heap

Definition: A priority queue is a data structure that allows elements to be removed based on priority (highest or lowest priority element is removed first).

Tasks:

- Implement a priority queue using a heap structure.
- Implement functions to insert elements with a priority and to remove the highest priority element.

Testing: Insert elements with varying priorities and demonstrate removing elements in priority order.

Instructions for Submission

1. Implement the above tasks in C, ensuring each function works as expected.
2. Capture the output for each function (tree traversal and heap sort).
3. Document each step and observation.
4. Submit a PDF containing the following:
    - C Code: Include all implemented code sections.
    - Output Screenshots: Attach screenshots of the code output for each function.
    - Explanation: Provide explanations for each step of the code, including observations and results.