Assignment 3

Submission Date: 26/09/2025

Q1. Bubble Sort

Write a program to implement Bubble Sort on an integer array.

Test Cases:

- Input: [5, 2, 9, 1, 5, 6] → Output: [1, 2, 5, 5, 6, 9]
- Input: [3, 2, 1] → Output: [1, 2, 3]
- Input: [1, 2, 3] → Output: [1, 2, 3]

Q2. Insertion Sort

Implement Insertion Sort to arrange elements in ascending order.

Test Cases:

- Input: [12, 11, 13, 5, 6] → Output: [5, 6, 11, 12, 13]
- Input: [4, 3, 2, 10, 12] → Output: [2, 3, 4, 10, 12]

Q3. Selection Sort

Sort an array using Selection Sort.

Test Cases:

- Input: [64, 25, 12, 22, 11] → Output: [11, 12, 22, 25, 64]
- Input: [29, 10, 14, 37, 13] → Output: [10, 13, 14, 29, 37]

Q4. Merge Sort

Write a recursive program to implement Merge Sort.

Test Cases:

- Input: [38, 27, 43, 3, 9, 82, 10] → Output: [3, 9, 10, 27, 38, 43, 82]
- Input: [5, 4, 3, 2, 1] → Output: [1, 2, 3, 4, 5]

Q5. Quick Sort

Implement Quick Sort using the last element as the pivot.

Test Cases:

- Input: [10, 7, 8, 9, 1, 5] → Output: [1, 5, 7, 8, 9, 10]
- Input: [1, 1, 1, 1] → Output: [1, 1, 1, 1]

Q6. Sorting Strings (Lexicographic Order)

Write a program to sort an array of strings using any sorting algorithm.

Test Cases:

- Input: ["apple", "banana", "cherry", "date"] → Output: ["apple", "banana", "cherry", "date"]

- Input: ["dog", "cat", "elephant", "bee"] → Output: ["bee", "cat", "dog", "elephant"]

Assignment: Tree Data Structures (Coding)

Q7. Create a Binary Tree & Print Preorder Traversal

Write a program to create a binary tree and print its preorder traversal.

Test Case:

Tree:

Expected Output (Preorder): 1 2 4 5 3

Q8. Inorder, Preorder, Postorder Traversals

Implement recursive functions to print inorder, preorder, and postorder traversals.

Test Case:

Tree:

- Inorder: 40 20 50 10 30
- Preorder: 10 20 40 50 30
- Postorder: 40 50 20 30 10

Q9. Level Order Traversal (BFS)

Implement level-order traversal of a binary tree.

Test Case:

Tree:

Output: 1 2 3 4 5 6 7

Q10. Insert into a BST
Write a program to insert nodes into a BST and print inorder traversal.
Test Case:
Insert: 50, 30, 20, 40, 70, 60, 80
Tree (BST) Inorder: 20 30 40 50 60 70 80

Q10. Search in BST
Write a program to search for a value in a BST. Return true if found, false otherwise.
Test Case:
BST from Q4 → Search for 40 → Output: Found
BST from Q4 → Search for 90 → Output: Not Found

Q11. Minimum & Maximum Node in BST
Find the smallest and largest element in a BST.
Test Case:
BST from Q4 → Min: 20, Max: 80

Q12. Height of Binary Tree
Write a recursive function to find the height of a binary tree.
Test Case:
Tree:

Height = 3
Q13. Check if a Binary Tree is Balanced
A balanced tree means the height difference of left and right subtrees for every node is ≤ 1.
Test Cases:

- Input: Balanced tree → Output: True
- Input: Skewed tree (like linked list: 1→2→3→4) → Output: False

Q14. Convert Sorted Array to Balanced BST
Write a program to convert a sorted array into a balanced BST.

Test Case:
Input: [1, 2, 3, 4, 5, 6, 7]
Output (Preorder example): 4 2 1 3 6 5 7