1st Sem SY 2025-2026

Name: Gabriel Cabrera	Program: BS Computer Engineering
Course: CPE 010	Professor: Engr. Jimlord Quejado
Section: CPE21S4	Date: September 30, 2025
Data Structures and Algorithms	Implementing Trees

Answer the following questions:

1. Define what is a binary tree?

A binary tree is a hierarchical data structure where each node has at most two children, referred to as the left child and the right child. It's like a family tree where each person can have up to two direct descendants. The topmost node is called the root, and nodes with no children are called leaves. Binary trees are used for tasks like searching, sorting, and organizing data efficiently due to their recursive structure.

2. What are the key components of a Binary tree?

A binary tree's key components include nodes, each holding data and pointers to up to two children, one each on the left and right. The root is the topmost node, serving as the starting point. Leaves are nodes with no children, marking the tree's endpoints. Parent nodes have at least one child, and subtrees are smaller binary trees formed by a node and its descendants, creating the hierarchical structure essential for organizing data.

3. What are the operations that can be performed in a Binary tree?

Binary tree operations include insertion, which adds a new node while preserving the tree's structure, something like adding a relative to a family tree, deletion, which removes a node and may reorganize the tree; and traversal, which visits nodes in orders like inorder (left, node, right), preorder (node, left, right), or postorder (left, right, node). Other operations involve searching for a specific value by navigating the tree and calculating the tree's height, which measures the longest path from root to leaf, useful for checking balance.

4. What are the conditions for a Binary Search Tree?

A Binary Search Tree is a binary tree where each node holds a key, and all keys in the left subtree are smaller than the node's key, while all keys in the right subtree are larger, like organizing books on a shelf by page count. Both subtrees must also be BSTs, and duplicate keys are not allowed. This structure enables efficient searching, insertion, and deletion, with operations taking O(h) time, where h is the tree's height, ideally O(log n) for balanced trees but O(n) for unbalanced ones.

5. Give a simple sample program in C++ that uses the above algorithm. Explain how the program works.

main.cpp

Share Run

```cpp
#include <iostream>
using namespace std;

struct Node {
    int data;
    Node* left;
    Node* right;
    Node(int val) : data(val), left(nullptr), right(nullptr) {}
};

class Binary {
private:
    Node* root;

    Node* insertNode(Node* node, int val) {
        if (node == nullptr) {
            return new Node(val);
        }
        if (val < node->data) {
            node->left = insertNode(node->left, val);
        } else if (val > node->data) {
            node->right = insertNode(node->right, val);
        }
        return node;
    }

    void inorderTraversal(Node* node) {
        if (node != nullptr) {
            inorderTraversal(node->left);
            cout << node->data << " ";
            inorderTraversal(node->right);up
        }
    }

public:
```

```
            inorderTraversal(node->left);
            cout << node->data << " ";
            inorderTraversal(node->right);up
        }
    }

public:
    Binary() : root(nullptr) {}

    void insert(int val) {
        root = insertNode(root, val);
    }

    void displayInorder() {
        cout << "Sorted Binary Search Tree (Inorder Traversal): ";
        inorderTraversal(root);
        cout << endl;
    }
};

int main() {
    Binary tree;
    int values[10];

    cout << "Enter 10 integers to insert into the Binary Search Tree: ";
    for (int i = 0; i < 10; i++) {
        cin >> values[i];
        tree.insert(values[i]);
    }

    tree.displayInorder();

    return 0;

}
```

Output Clear

```
Enter 10 integers to insert into the Binary Search Tree: 10 20 14 59 20 45 32 69 7 88
Sorted Binary Search Tree (Inorder Traversal): 7 10 14 20 32 45 59 69 88


=== Code Execution Successful ===
```

The program defines a Node structure containing an integer value and pointers to left and right children, forming the building blocks of the tree. The Binary class manages the Binary Search Tree, with a private root pointer initialized to nullptr. The insertNode function recursively adds a new value: if the current node is nullptr, it creates a new node; otherwise, it compares the value to the node's data, navigating left for smaller values or right for larger ones, ensuring the BST property (smaller keys on the left, larger on the right). The inorderTraversal function visits nodes recursively in the order: left subtree, current node, right subtree, printing values in ascending order. In the main function, the program prompts the user to input 10 integers, inserts them into the tree, and displays the sorted values using inorder traversal.