National University of Computer and Emerging Sciences

Assignment No. 3

Data Structures

Trees (BST & AVL)

CS2001

Fall 2024

Deadline: October 31, 2024

Submission Instructions:

- All problems must be solved by following the order and submitted as one .zip file.
- All Problems must be in three file Structure.
- All Problems must be based on user defined Inputs.
- Parent folder should be named as **DS_ASS03_XXF_YYYY**, where XX represents the batch and YYYY represents the roll number. Compress this folder as **DS_ASS03_XXF_YYYY.zip** and submit this one file.
- Also submit a **.doc file** containing all the codes along with their **screenshots**.
- Any submission not following the submission instructions will not be evaluated.
- This is an individual assignment.
- **Plagiarism is strictly prohibited.**

Question 01: BST ADT [Marks 10]

Implement the following functions for the Binary Search Tree:

a) **Function to insert a node in BST.**

b) **Function to search a specific node in BST.**

c) **Function to display all the nodes.**

```
class Node {
public:
        int data;
        Node* l_Child;
        Node* r_Child;
        Node(int);
};
class BST {
private:
        Node* root;
public:
        BST();
```

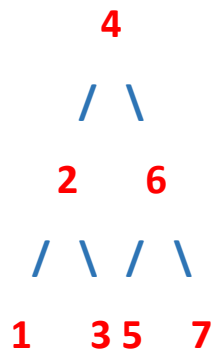
```
        Node* insertion(Node*, int);
        bool find(Node*, int);
        void inOrder(Node*);
        void preOrder(Node*);
        void postOrder(Node*);
};// you can add methods as per requirement
```

Question 02: BST With Minimal Height [Marks 30]

Implement a function that converts a sorted array to a BST of minimal height.

For Example :

Sorted Array : [1, 2, 3, 4, 5, 6, 7]

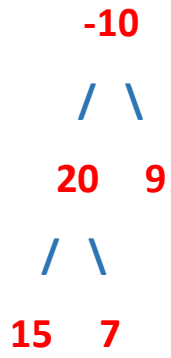


Question 03: Find maximum sum in BST [Marks 20]

A path in a binary tree is a sequence of nodes where each pair of adjacent nodes in the sequence has an edge connecting them. A node can only appear in the sequence at most once. Note that the path does not need to pass through the root.
The path sum of a path is the sum of the node's values in the path.
Given the root of a binary tree, return *the maximum path sum of any non-empty path*.

For Example 1:

Input: root =
[-10,9,20,null,null,15,7]
Output: 42
Explanation: The optimal path is 15 -> 20 -> 7 with a path sum of 15 + 20 + 7 = 42.



Example 2:

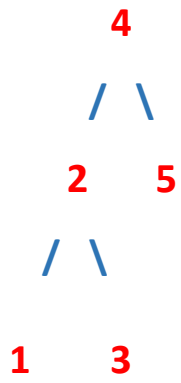
Input: root = [1,2,3]
Output: 6
Explanation: The optimal path is 2 -> 1 -> 3 with a path sum of 2 + 1 + 3 = 6.

Question 04: BST to Sorted Doubly Linked List [Marks 30]

You are given the root of a Binary Search Tree (BST). Convert the BST to a circular doubly linked list in-place. The left and right pointers in a node are to be used as the previous and next pointers respectively in the doubly linked list. The order of the nodes in the doubly linked list must follow the in-order traversal of the binary search tree, and the linked list must be circular—meaning the tail node's next pointer should point to the head of the list, and the head node's previous pointer should point to the tail.

For Example:

- **Input Tree**



- **Output**

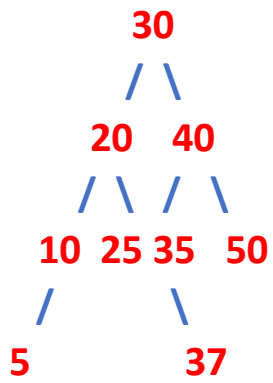
A circular doubly linked list: **1 <-> 2 <-> 3 <-> 4 <-> 5**

Question 05: K-th Smallest [Marks 20]

You are working on a search engine optimization project for a large e-commerce platform. The platform uses a Binary Search Tree (BST) to manage and retrieve product prices efficiently. To enhance the user experience, the platform needs to display the kth smallest price among the available products.

For Example:

- **Input Tree**



1. **For k = 1, the function should return 5.**

2. **For k = 4, the function should return 25.**
3. **For k = 6, the function should return 35**

Question 06: AVL ADT Using Templates [Marks 20]

Provide a C++ implementation of AVL tree using Templates that must include:

- **Recursive Height**
- **Finding Balancing Factor**
- **RR_Rotation**
- **LL_Rotation**
- **RL_Rotation**
- **LR_Rotation**
- **AVL node deletion**
- **AVL node insertion**
- **Display Nodes**

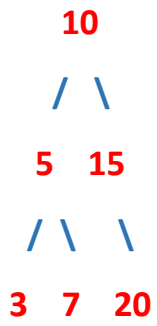
Test All Functions . It must Run on All user defined Inputs .

Question 07: Merge Two AVL Trees [Marks20]

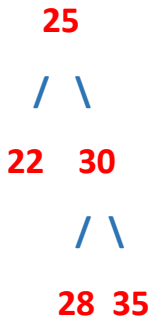
You are a software engineer working for a company that handles large volumes of financial transactions. To manage and balance these transactions efficiently, the company uses AVL trees due to their self-balancing properties. You are tasked with merging two AVL trees, representing different subsets of transaction amounts, into a single AVL tree while maintaining the AVL tree properties.

For Example:

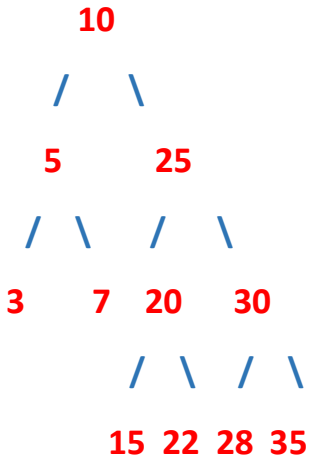
Input Tree 1



Input Tree 2



Output:



Hint : Merge Second AVL Tree into First AVL Tree .

Question 08: Most Frequent Songs [Marks 30]

You have recently joined Spotify Technology S.A. as a software engineer, and your task is to design an algorithm and develop a menu-driven program that efficiently stores and

manages the most frequently played songs in an AVL tree. Each node in the AVL tree should contain the song name and its frequency, and the tree should be constructed based on the song frequency. Additionally, the program should allow users to play songs.

Once a song is played, its frequency will increase by 1, ensuring the tree remains balanced. If two or more songs have the same frequencies, then add songs with the same frequency next to each other by creating a linked list of nodes in the tree.

To get you started, you are provided with the following initial song frequencies:

Song_A: Frequency = 1

Song_B: Frequency = 5

Song_C: Frequency = 9

Song_D: Frequency = 2

Song_E: Frequency = 4

Song_F: Frequency = 6

Song_G: Frequency = 8

Song_H: Frequency = 3

Song_I: Frequency = 7

Song_J: Frequency = 9

Song_K: Frequency = 5

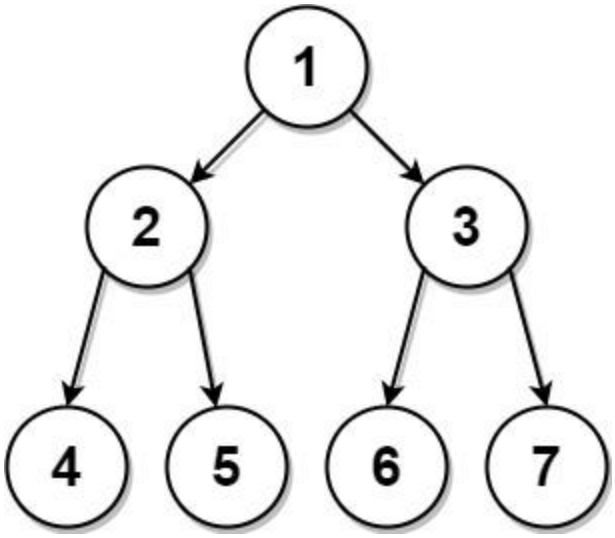
Question 9: Construct BST using preorder and post order Traversal

[Marks 20]

Given two integer arrays, preorder and postorder where preorder is the preorder traversal of a binary tree of distinct values and postorder is the postorder traversal of the same tree, reconstruct and return *the binary tree*.

If there exist multiple answers, you can return any of them.

For Example :



Input: preorder = [1,2,4,5,3,6,7], postorder = [4,5,2,6,7,3,1]

Output: [1,2,3,4,5,6,7]

Example 2:

Input: preorder = [1], postorder = [1]

Output:[1]

The **magic** you are looking for is in the work you are avoiding. 😊