Data Structures and Algorithms
LAB-10

Total Marks: 50

Start Time: 1:45 PM
CompletionTime: 3:20 PM

Objective:

In this lab, students will practice about the working of Binary Trees.

Instructions:

1) Follow the question instructions very carefully, no changes in function prototypes are allowed.
2) Your laptops must be on airplane mode
3) Anyone caught in an act of plagiarism would be awarded an "F" grade in this Lab.

TASK-01: Auto-Complete using BST [20 Marks]

You are given a sorted list of words. Your task is to build a Binary Search Tree (BST) from these words, and implement an auto-complete feature. In this feature, when a user enters a prefix, the system should return all words stored in the BST that start with that prefix.

Constraints

- All words are lowercase alphabetic strings.
- The BST must be constructed using standard lexicographical BST rules.
- Prefix can be any valid lowercase string (even empty).

Implement:

- A function to convert `list of words` to a BST
  Node * wordsToBST(const vector<string>& words)

- A function to return all words with a given prefix from that BST.
  vector<string> autoComplete(Node* root, const string& prefix)

Input:

- A list of lowercase words (sorted or unsorted).
- A string prefix.

Output

- A list of all words from the BST that start with the given prefix, in **any order**.

Example 1:

Input:

words = ["apple", "appetizer", "banana", "ball", "bat", "battle", "cat"]

prefix = "ba"

Output :

["banana", "ball", "bat", "battle"]

Example 2:

Input:

words = ["apple", "appetizer", "banana", "ball", "bat", "battle", "cat"]

prefix = "app"

Output :

["apple", "appetizer"]

TASK-02: News Feed Aggregator (K-th Most Viewed Post) [15 Marks]

You are building a news feed aggregator system. Each post is represented as a node in a Binary Search Tree (BST) where:

- The key of each node is the number of views on that post.
- No two posts have the same number of views.

Your task is to write a function that returns the k-th most viewed post in the BST.
Here, 1st most viewed means the post with the highest number of views, 2nd most viewed means second-highest, and so on.

Input

- The root of a BST, where each node contains a unique integer representing the view count.
- An integer k.
- Prototype: ***int kthMostViewedPost(Node* root, int k)***

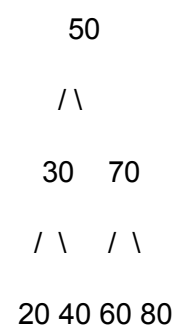
Output

- The view count of the **k-th most viewed post**.

Constraint:

- Do not use any extra data structures(e.g, arrays, vectors)

BST Structure (view counts):



Example1:

Input: k = 5

Output : 40

Explanation: Descending order: 80 (1st), 70 (2nd), 60 (3rd), 50 (4th), 40 (5th)

Example2:

Input: k = 2

Output : 70

Explanation: Descending order: 80 (1st), 70 (2nd), 60 (3rd), ...

So 2nd most viewed = 70

TASK-03: Path Sum in a Binary Tree [15 Marks]

You are given the **root of a binary tree** and an integer targetSum.
Your task is to determine whether there exists **at least one root-to-leaf path** in the tree such that the **sum of all the node values** along the path is equal to targetSum.A **leaf** is a node with no children.

Input

- A binary tree where each node contains an integer value.
- An integer targetSum — the required sum to check for.

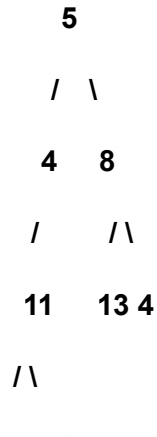
Output

Return true if such a path exists, otherwise return false.

Function Prototype:

bool hasPathSum(TreeNode* root, int targetSum)

Tree Structure:



Input: targetSum = 22 , Output: true

Explanation:

Path 5 → 4 → 11 → 2 has a sum of 22.