National University of Computer and Emerging Sciences, Lahore Campus

Course Name:	**Data Structures**	**Course Code:**	**CL2001**
Program:	**BS(CS)**	**Semester:**	**Fall 2023**
Duration:	**1.0 Hours**	**Total Marks:**	**50**
Paper Date:	**17-Nov-23**	**Weight:**	**7.5%**
Section:	**BSC-3F**	**Pages:**	**2**
Exam:	**Quiz-2**		

Student : Name:_ **Roll No.________________** **Section:_**

Instruction/Notes:

Problem: BST Erase Function with Scenarios

You are working on a library system, and part of the system involves managing a collection of books using a Binary Search Tree (BST). Each book in the BST is represented by its unique ISBN (International Standard Book Number), and you need to implement the erase function to handle different scenarios.

Book with No Children:

- If the book to be removed has no children, simply remove it from the tree.

Book with One Child:

- If the book has one child, replace the book with its child.

Book with Two Children:

- If the book has two children, find the book's in-order successor (or predecessor).
- Replace the book's data with the in-order successor's (or predecessor's) data.
- Remove the in-order successor (or predecessor).

Problem: BST with Height Calculation

You are working on a library system, and part of the system involves managing a collection of books using a Binary Search Tree (BST). Extend your existing BST implementation with a recursive method called height that computes the height of the BST. The height of a node is one more than the height of its taller child.

Problem: BST with Depth Calculation

Extend your existing BST implementation with a method called depth that calculates and returns the depth of a specified key in the BST. The depth of a node (or key) is the number of edges on the path from the root to that node. The method should take O(h) time, where h is the height of the BST.

Constraints:

The ISBNs are unique strings with a maximum length of 20 characters.

The title and author strings have a maximum length of 50 characters.

Problem: BST with Balanced Check

Extend your existing BST implementation with a method called isBalanced that checks if the BST is balanced or not. A balanced tree is defined as a tree in which the difference in the heights of the two subtrees of every node is no more than 1. The balance factor (bf) of a node x is calculated as bf(x) = height(x.right) - height(x.left).

Constraints:

The ISBNs are unique strings with a maximum length of 20 characters.

The title and author strings have a maximum length of 50 characters.

Problem: Parameterized Constructor for Perfect BST

Extend your existing BST implementation with a parameterized constructor that converts the given sorted data array into a perfect (or complete) BST. The constructor should have the signature bst(T sortedData[], int n) and should have a running time of no more than O(n).

Constraints:

The T type parameter can be any comparable data type (e.g., int, string).

The size n of the sortedData array is in the range [0, 10^5].

Problem: Path Sums in BST

Extend your existing BST implementation with a method called pathSums that returns a vector containing the sums of the keys along each path of the tree, starting from the root to each leaf.

Constraints:

The T type parameter can be any numeric data type (e.g., int, float, double).

The keys in the BST are unique.