CS2001 **– Data Structures**

Assignment # 03

Instructor	Mr. Muhammad Yousaf
Session	Fall 2022
Section	BCS 3D – MCS

General Guidelines

1. Write neat and clean code. Avoid any memory leaks and dangling pointers while implementing the scenarios required in this assignment.
2. You can lose the marks if conventions are not strictly followed.
3. Peer plagiarism and the late submissions are strictly not allowed. In case, zero marks will be awarded for whole assignment. You're not allowed to use any built-in libraries.
4. Total Marks: 250

Submission Guidelines

1. You will upload the assignment on CLASSROOM in given timeline.
2. Don't email your solution to instructor or TA for submission. Submit your assignment in given deadline said LMS.
3. You have already given one extra day for submission. No submission will be accepted later than said deadline.
4. Set file name as *Roll-no_Section_Assignment#*

Deadline: October 31, 2022, 4:00 PM

Binary Tree Implementation

Task # 1: Tree ADT [Marks 60]

Consider the following functions to implement the Tree ADT.

Insert(): insert an element in the tree

DeleteKey(): delete an element in the tree

SearchKey(): searches the desired element in the tree recursively

findMax(): finds the maximum element in the tree recursively

findMin(): finds the minimum element in the tree recursively

inorderTraversal() : prints in-order traversal of the tree

preorderTraversal(): prints pre-order traversal of the tree

postorderTraversal(): prints post-order traversal of the tree

treeHeight(): returns the height of the tree recursively

treeNodeCount(): returns the count of nodes in the tree

treeLeavesCount(): returns the count of leaves in the tree

printNodeLevel(): prints level of a node in the tree

The class will be created in a file "binarySearchTree.h" and the functions will be implemented in "binarySearchTree.cpp". Make sure to create the class as a template so it can run for multiple data types.

Your main function should like: `int main(){`
```
// Sample Input: 65 55 22 44 61 19 90 10 78
52
BST<int> myIntBST;
// test all functions here
}
```

Task # 2: Mirror Binary Tree [Marks 20]

Write a recursive function that converts the given binary tree into its mirror tree. **Hint**: The mirror of a binary tree T is another binary Tree M(T) with left and right children of all ***non-leaf nodes*** interchanged. Following attached diagram depicts the mirror trees however, you are required to consider only non-leaf nodes for the implementation of this task.
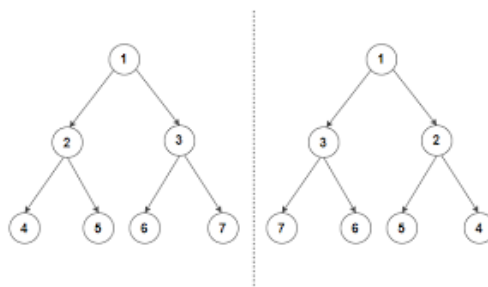
Task # 3: BST with Minimal Height [Marks 20]

Convert a sorted array to a BST of minimal height (Use the library you implemented in task 1)

Task # 4: Binary Binary Tree [Marks 10+20]

a. Write a recursive function that will calculate the sum of all nodes given in binary tree
b. Consider a binary tree with binary values (1s and 0s) in the nodes with same insertion rules as BST. However, in this task, you can consider the recurring value 1 is greater than previous 1. Similarly, 0 can be considered as less than next consecutive zero. After insertion of the given sample inputs, you're required to apply the logical AND, OR, and XOR gates between the nodes in reverse polish fashion. For example, the BST for input pattern (101) can be constructed as follow:

$$1$$
$$/ \ \backslash$$
$$0 \quad 1$$

Output for the tree for AND is 0 and XOR 0. The truth tables can be referred to as

Name	NOT		AND			NAND			OR			NOR			XOR		
Alg. Expr.	$\overline{A}$		AB			$\overline{AB}$			$A+B$			$\overline{A+B}$			$A \oplus B$		
Symbol																	
Truth Table	A	X	B	A	X	B	A	X	B	A	X	B	A	X	B	A	X
	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
	1	0	0	1	0	0	1	1	0	1	1	0	1	0	0	1	1
			1	0	0	1	0	1	1	0	1	1	0	0	1	0	1
			1	1	1	1	1	0	1	1	1	1	1	0	1	1	0

Task # 5: Binary Search Tree Applications [Marks 20 + 10]

Question 1: Given an array nums that represents a permutation of integers from 1 to n. We are going to construct a binary search tree (BST) by inserting the elements of nums in order into an initially empty BST. Find the number of different ways to reorder nums so that the constructed BST is identical to that formed from the original array nums.

For example, given nums = [2,1,3], we will have 2 as the root, 1 as a left child, and 3 as a right child. The array [2,3,1] also yields the same BST but [3,2,1] yields a different BST. Return *the number of ways to reorder* nums *such that the BST formed is identical to the original BST formed from* nums.

```
Input: nums = [3,4,5,1,2] Output: 5
[3,1,2,4,5]
[3,1,4,2,5]
[3,1,4,5,2]
[3,4,1,2,5]
[3,4,1,5,2]
```

Question 2: Given the root of a Binary Search Tree (BST), convert it to a Greater Tree such that every value of the original BST is changed to the original value plus the sum of all nodes greater than the current in BST.

As a reminder, a binary search tree is a tree that satisfies these constraints:

- The left subtree of a node contains only nodes with values less than the node's value.
- The right subtree of a node contains only nodes with values greater than the node's value.
- Both the left and right subtrees must also be binary search trees.