CS2001 – Data Structures

Assignment # 04

Instructor	**Mr. Muhammad Usman Joyia**
Session	**Fall 2022**
Section	**BCS 3A – BSE 3A**

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: 100

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: Friday, November 11, 2022, 4:00 PM

AVL Tree Implementation

Task # 1: AVL Tree ADT [Marks 60]

You have already implemented the following Tree ADT in assignment 3. Now, consider the following functions again to implement the AVL Tree.

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 "AVLTree.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. In addition to the methods in task 1, you also must implement methods that keep the tree balanced.

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: 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 # 3: Sum Pairs [Marks 20]

Find a pair with a given sum in AVL tree using the library you implemented in task 1.