National University of Computer and Emerging Sciences

Lab Manual

for

Data Structures

Course Instructor	Ma'am Abeeda Akram
Lab Instructor(s)	Mr. Sohaib Ahmad Ms. Ammara Nasir
Section	BCS-3G
Semester	FALL 2022

Department of Computer Science
FAST-NU, Lahore, Pakistan

Lab Manual 04

Objectives:

After performing this lab, students shall be able to revise:

- ✓ Delete operation on BST
- ✓ Insert and Delete operation on AVL Trees

Question1:

```
struct Node
{
	int data;
	Node *left;
	Node *right;
};
class BST
{
	Node* root;
};
```

Add the following two member functions for your binary search tree class implemented in last lab :

1) bool delete_ recursive (int val) // Write recursive delete function for BST. Delete a node based on given value. Note that in deletion, you have to deal with 3 cases.
    a. Node to be deleted is a leaf node (No Children).
    b. Node to be deleted has only one child.
    c. Node to be deleted has two children.

Question #2

Now create a node as follows for AVL and implement class AVL which contains a pointer to the root node of AVL tree of type.

```
struct AVLNode
{
        int data;
        AVLNode *left;
        AVLNode *right;
        int hl;  // left height
        int hr;  // right height

//implement any required constructors/getters/setters.
};
class AVL

{

        AVLNode * root;

};
```

In AVL ,we maintain balance by ensuring the difference in height of left subtree and right subtree is never more than 1. Implement the following methods for AVL class:

NOTE: Use helper functions if required.

1. AVL(); //default constructor
2. void rightRotate(AVLNode *& x) // This function performs right rotation
3. void leftRotate(AVLNode *& x) // This function performs left rotation
4. void balanceCheck(AVLNode * & t) // This function updates the height of nodes after performing rotations
5. void insert(int data); //inserts a new node in the AVL tree and maintains the height property of the AVL tree.
   Note:
   - First you will simply perform insertion in the same way you used to do in your BST
   - Before concluding the insertion operation, you will check whether the tree is Balanced or not
   - In case of an imbalance (In AVL the difference of height between both sub trees should not be more than 1), you have to perform rotation operations till the tree is balanced.

6. void deletedata (int data); //deletes the node containing the data, and maintains the height property of AVL tree.

7. void Printintreeform(); //prints the data in Preorder

8. ~AVL() // destructor

Now run the following main program

```
int main()
{
       AVL tree;
       tree.insert(10);
       tree.insert(20);
       tree.insert(30);
       tree.insert(40);
       tree.insert(50);
       tree.insert(25);
       tree. Printintreeform ();
//Output of Preorder traversal 30 20 10 25 40 50

      system("pause";

      return 0;

}
```

Output in tree form should be like this after inserting all the given elements.

```
                    30
                   /   \
                20     40
               /    \       \
            10     25     50
```

Good Luck!