National University of Computer and Emerging Sciences

Laboratory Manual

for

Data Structures Lab

Course Instructor	Miss Arooj Khalil
Lab Instructor(s)	Mr. Dilawar Shabbir Mr. Sohaib Ahmad
Section	SE-3A
Semester	Fall 2022

Department of Computer Science

FAST-NU, Lahore, Pakistan

Objectives:

In this lab, students will practice:

1. BST
2. AVL

Tree

visualization: https://www.cs.usfca.edu/~galles/visualization/BST.html

Implement the following Tree Node:

```
struct Node
{
        int  data;
        Node*left;
        Node *right;
};
```

Problem 1

Implement a binary search tree class "BST" which contains the root of type **Node** as a data member.

```
class BST
{
        Node* root;
};
```

Implement the following member functions for BST:

NOTE: Use helper functions if required.

1. A default Constructor which sets the root to nullptr.
2. Implement a function 'insert'. It should insert the data while considering the insertion rules. If the data already exists in the BST, simply return false and true otherwise.
   `bool insert(int v)`
3. A copy constructor which uses recursion to deep copy another Binary Search Tree object.
4. A function "inorderPrint" prints the keys using in-order traversal.
   `void inorderPrint () const`
5. Use level order traversal for the printing of trees, level by level.
   `void levelorderPrint () const`
6. A function "search". The function then uses recursion to return a pointer to the corresponding node. If the key does not exist, the function returns nullptr.
   `Node* search(int key)`
7. Use inorder LVR to implement a recursive function "countNodes" to return the count of total nodes in BST.
   `int countNodes() const`
8. Use Preorder traversal VLR to implement a recursive function "leafCount" to return the count of leaf nodes in BST.

int leafCount() const

9. Use Postorder LRV to implement the Destructor for BST.

Problem 2

visualization: https://www.cs.usfca.edu/~galles/visualization/AVLtree.html

Implement a self-balancing tree AVL the structure of the class is given below. you can add helper functions as private data members of the class if required.

```
class AVL
{
        Node* root;
        int getBalance(Node *node);
        Node* insertHelper(Node* node, int key);
        Node *leftRotate(Node *node);
        Node *rightRotate(Node *node);
public:
        void insert(int key);
        void preOrderPrint() const;
};
```