**National University of Computer and Emerging Sciences**



**Laboratory Manual**

*for*

# Data Structures Lab

| Course Instructor | Mr. Uzair Naqvi |
| --- | --- |
| Lab Instructor(s) | Ms. Fariha Maqbool  Ms. Marwa Khan |
| Section | BCS-3H |
| Semester | Fall 2022 |

# Department of Computer Science

FAST-NU, Lahore, Pakistan

**Objectives:**

In this lab, students will practice:

1. Binary Search Trees
2. Recursive insert operation, Preorder, Inorder and Postorder traversal, recursive Search operation, Tree Length and Count of Leaf nodes operations on BST

**Question 1:**

Implement the following Tree Node:

struct Node

{

int data;

Node\*left;

Node \*right;

};

Now implement a binary search tree class “BST” which contains root of type **Node** as data member.

class BST

{

Node\* root;

};

You have to implement the following member functions for your binary search tree:

1. A default Constructor which sets the root to nullptr.

1. A recursive “insert” function which is passed as parameter **int data**. 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 d)

1. A copy constructor which uses recursion to deep copy another Binary Search Tree object.

1. A function “inorderPrint” which prints the keys using pre-order traversal.

void preorderPrint () const

1. A function “inorderPrint” which prints the keys using in-order traversal.

void inorderPrint () const

1. A function “inorderPrint” which prints the keys using post-order traversal.

void postorderPrint () const

1. A function “search” which is passed as parameter a key. The function then uses recursion to return pointer to the corresponding node. If the key does not exist, the function returns nullptr. Node\* search(int key)

1. A function “length” which uses recursion to return the count of total nodes in BST.

int length() const

1. function “leafCount” which uses recursion to return the count of leaf nodes in BST.

int leafCount() const

1. Destructor

**Your MAIN FUNCTION:**

void main()

{

BST<int> tree;

tree.insert (500); tree.insert (1000); tree.insert (1); tree.insert (600); tree.insert (700); tree.insert (10); tree.insert (30); tree.insert (9000); tree.insert (50000); tree.insert (20);

cout << "Printing data using recursive inorder traversal: "; tree.InorderPrint ();

cout << "\nCOPY CONSTRUCTOR\n";

BST<int> tree1(tree);

cout << "\n Preorder Traversal \n"; tree1.preorderorderPrint (); cout << "\n Inorder Traversal \n";

tree1.inorderPrint (); cout << "\n Postorder Traversal \n"; tree1.postorderPrint ();

cout << "\n SEARCH: ";

tree.search(1); tree.search(30); tree.search(50);

cout << "\n Tree Length: " << tree.length() << endl << endl;

cout << "\n Tree Leaf Nodes: " << tree.leafCount() << "\n";

cout << endl; system("pause");

}