

National University of Computer and Emerging Sciences



Lab Manual
for
Data Structure

Course Instructor	Ms. Zareen Alamgir
Lab Instructor(s)	Ms. Mamoonah Akbar Ms. Ammara
Section	BCS-3C
Semester	FALL 2022

Department of Computer Science
FAST-NU, Lahore, Pakistan

Lab Manual 07

Objectives:

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

- ✓ Binary Search Tree
- ✓ Recursive and iterative insert operation and some other operation on BST

Problem 1

Given Node Class

```
template <class T>
Class Node
{
    T data;
    Node<T> *left;
    Node<T> *right;
    Node(T d = 0);
}
template <class T>
Node <T>::Node(T d) {
    data = d;
    left = NULL;
    right = NULL;
}
```

Now implement class tree, which is Binary Search Tree and has node type **root** as a data member. In the class BST, you have to implement the following functions.

1. `BST()` // Default constructor, which sets root to NULL
2. `BST(const BST & obj)` // Write a copy constructor for BST, which must create a deep copy. Note that copy constructor will make the copy recursively.
3. `void insert(T n)` // It iteratively inserts the value while considering the insertion rules for Binary Search Tree.
 - To insert a node, our first task is to find the place to insert the node.
 - Take `current = root`.
 - start from the current and compare root data with `n`
 - if the current data is greater than `n`, which means we need to go to the left of the root.
 - if the current data is smaller than `n`, we need to go to the right of the root.
 - if any point of time current is null, that means we have reached to the leaf node, insert your node here with the help of the parent node.
4. `bool find(T value)` // This function returns true if given value is found otherwise false.

5. `bool delete_iterative(T val)` // Delete a node based on given value. In deletion, you have to deal with 3 cases.
 - Node to be deleted is a leaf node (No Children).
 - Node to be deleted has only one child.
 - Node to be deleted has two children.
6. `void insert_recursive(T value, Node<T> *& node)` // Write a recursive function to insert a value in BST where we examine the root and recursively insert the new node to the left subtree if its key is less than that of the root or the right subtree if its key is greater than or equal to the root.
7. `bool delete_recursive(T val)` // Write recursive delete function
8. `void display()` // Print the entire tree in increasing order. This should be recursive.
9. `void leafSum(Node<T>*root, int& sum)` // Implement a recursive function void leafSum() to find total leaf nodes of BST.
10. Write a destructor to clean the memory. When deleting a node, first delete the left and right child and then the node itself.