**National University of Computer and Emerging Sciences**



Laboratory Manual

for

Data Structures Lab

|  |  |
| --- | --- |
| Course Instructor | Samman Ashraf |
| Lab Instructor(s) | Muhammad Ahmed  Hamza Ayub |
| Section | 4N |
| Semester | Spring 2023 |

**Department of Computer Science**

FAST-NU, Lahore, Pakistan

**Objectives:**

In this lab, students will practice:

1. Binary Search Trees
2. Recursive insert operation, recursive delete operation, and inorder traversal, and some other recursive operations on BST

**Question 1**

Implement the following Tree Node:

template<typenamek, typenamev>

structTNode

{

k key;

v value;

TNode<k, v> \*leftChild;

TNode<k, v> \*rightChild;

}

Now implement a binary search tree class “BST” which contains root of type TNode as data member. You have to implement the following member functions for your binary search tree:

* 1. A default Constructor which sets the root to nullptr.
  2. A copy constructor which uses recursion to deep copy another Binary Search Tree object.
  3. A recursive “insert” function which is passed as parameter a key and a corresponding value. It then inserts the <key, value> pair while considering the insertion rules. If the key already exists in the BST, simply replace the value.

void insert(k const key, v const value)

* 1. A function “search” which is passed as parameter a key. The function then uses recursion to return pointer to the corresponding value. If the key does not exist, the function returns null.

v\* search(k key)

* 1. A function “inorderPrintkeys” which prints the keys using inorder traversal.

void inorderPrintKeys() const

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

int length() const

* 1. A “deleteKey” function which is passed as parameter a key. The function then uses recursion to delete the node that contains that key.

void deleteKey(k key)

* 1. Destructor

**Question 2: Now run the following main program.**

int main()

{

BST<int, int> tree;//the key and value both are of type int

tree.insert(500, 500);

tree.insert(1000, 1000);

tree.insert(1, 1);

tree.insert(600, 600);

tree.insert(700, 700);

tree.insert(10, 10);

tree.insert(30, 30);

tree.insert(9000, 9000);

tree.insert(50000, 50000);

tree.insert(20, 20);

BST<int, int> tree1(tree);

tree1.deleteKey(20);

tree1.deleteKey(500);

tree1.deleteKey(1000);

tree1.inorderPrintKeys();

cout << endl << endl;

cout<<"Tree1 Length: "<<tree1.length()<<endl;

int \*val = tree1.search(1);

if (val != nullptr)

{

cout <<"1 found"<< endl;

}

val = tree1.search(123);

if (val == nullptr)

{

cout <<"123 not found"<< endl;

}

system("pause");

}

**Question No 3: Create a binary search tree from an array of integers.**

The program should prompt the user to enter the size of the array and then the elements of the array. The program should then create a binary search tree from the array and display the tree in pre-order, in-order, and post-order traversals.

Make sure to define a node struct with fields for the value, left child, and right child, as well as a class for the binary search tree that contains methods for inserting nodes and traversing the tree in pre-order, in-order, and post-order. You may assume that the array contains no duplicate elements.