**National University of Computer and Emerging Sciences**



Laboratory Manual

for

Data Structures Lab

|  |  |
| --- | --- |
| Course Instructor | Mr. Saad Farooq |
| Lab Instructor(s) | Mr. Hasnain Iqbal  Mr. Usama Hassan |
| Section | CS-E |
| Date | 29-Nov-2021 |
| Semester | Fall 2021 |

**Department of Computer Science**

FAST-NU, Lahore, Pakistan

**Objectives:**

In this lab, students will practice:

1. Binary Search Trees
2. Recursive insert operation, recursive inorder traversal, and some other recursive operations on BST
3. Iterative insert and Iterative inorder traversal using stack

**Question 1**

Implement the following Tree Node:

template <typename k, typename v>

struct TNode

{

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.

* 1. A recursive “insertRec” function which is passed as parameter a key and a corresponding value. It then uses **recursion** to insert the <key, value> pair while considering the insertion rules. If the key already exists in the BST, it simply replaces the value.

void insertRec(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 **recursive** inorder traversal.

void inorderPrintKeysRec() const

* 1. A function “preOrderPrintkeys” which prints the keys using **recursive** preOrder traversal.

void preOrderPrintKeys() const

* 1. A function “postOrderPrintkeys” which prints the keys using **recursive** postOrder traversal.

void postOrderPrintKeys() const

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

int length() const

* 1. A function “deleteNode” which is passed as parameter a key. The function then delete the node containing that key. If the node isn’t a leaf than replace the value with inorder predecessor.

void deleteNode(k const key) const

* 1. A function “inorderPredecessor” which is passed as parameter a pointer to node. The function then search the inorderPredecessor and return its value. node containing that key. If the node isn’t a leaf than replace the value with inorder predecessor.

v printAllAncestors(TNode \*) const

* 1. Provide the destructor which delete all the nodes from the tree
  2. bool isPerfect()

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

int main()

{

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

tree.insertRec(500, 500);

tree.insertRec(1000, 1000);

tree.insertRec(1, 1);

tree.insertRec(600, 600);

tree.insertRec(700, 700);

tree.insertRec(10, 10);

tree.insertRec(30, 30);

tree.insertRec(9000, 9000);

tree.insertRec(50000, 50000);

tree.insertRec(20, 20);

cout << "Printing keys using inorder traversal: ";

tree.inorderPrintKeys();

cout << endl << endl << "Printing keys using recursive inorder traversal: ";

tree.inorderPrintKeysRec();

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

int \*val = tree.search(123);

if (val != nullptr){

cout << "123 found" << endl;}

val = tree.search(123);

if (val == nullptr){

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

cout <<endl<< "Printing the keys using preOrder traversal: "; tree.preOrderPrintKeys();

cout <<endl<< "Printing the keys of ancestor nodes of 20";

tree.printAllAncestors(20);

tree.delete(1);

cout << "Post order traversal: "; tree.postOrderPrintKeys();

cout<<tree.isPerfect();

system("pause");

}