**National University of Computer and Emerging Sciences**



Laboratory Manual 06

for

Data Structures Lab

| Course Instructor | Dr. Saira Karim |
| --- | --- |
| Lab Instructor(s) | Ms. Fariha Maqbool  Ms. Muntaha Zaigham |
| Section | BCS-3E |
| Semester | Fall 2022 |

**Department of Computer Science**

FAST-NU, Lahore, Pakistan

**Objectives:**

In this lab, students will practice:

1. Recursion

**Question 1:**

Write a recursive function that takes an array that may contain more arrays in it and returns an array with all values flattened.

Suppose this is the input array:

[[1], [2, 3], [4], [3, [2, 4]]]

The output should be:

[1, 2, 3, 4, 3, 2, 4]

**Question 2:**

Given two matrices A and B, write a recursive function that returns the product of the two matrices.

**Question 3:**

**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:**

**a. A default Constructor which sets the root to nullptr.**

**b. 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)**

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

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

**void inorderPrint () const**

**e. 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)**

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

**int length() const**

**g. 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(int key)**

**h. Destructor**

**Your MAIN FUNCTION:**

**void main()**

**{**

**BST<int> tree;**

**tree.insertRec(500);**

**tree.insertRec(1000);**

**tree.insertRec(1);**

**tree.insertRec(600);**

**tree.insertRec(700);**

**tree.insertRec(10);**

**tree.insertRec(30);**

**tree.insertRec(9000);**

**tree.insertRec(50000);**

**tree.insertRec(20);**

**cout << "Printing datas using recursive inorder traversal: ";**

**tree.inorderPrintdatasRec();**

**cout << "\nCOPY CONSTRUCTOR\n";**

**BST<int> tree1(tree);**

**tree1.inorderPrintdatasRec();**

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

**cout << "DELETE KEY\n";**

**tree1.deleteKey(9000);**

**tree1.inorderPrintdatasRec();**

**cout << "\nSEARCH:";**

**if (tree.search(1))**

**{**

**cout << "\nFound" << endl;**

**}**

**else**

**cout << "NOT FOUND\n";**

**if (tree.search(123))**

**{**

**cout << "Found" << endl;**

**}**

**else**

**cout << "NOT FOUND\n";**

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

**cout << endl;**

**system("pause");**

**}**