University of Central Punjab

**Faculty of Information Technology**

# Data Structures and Algorithms

|  |  |  |
| --- | --- | --- |
|  | |  |
| **Topic** | * BST |
| **Objective** | * The basic purpose of this lab is to learn and implement BST. |
|  | | |

**Instructions:**

* Indent your code.
* Comment your code.
* Use meaningful variable names.
* Plan your code carefully on a piece of paper before you implement it.

**Students are required to complete the following tasks in lab timings.**

## Task 1

A tree whose elements have at-most two children is called a **Binary Tree.**

**Binary Search Tree** is a node-based binary tree data structure which has the following properties:

* The left subtree of a node contains only nodes with keys lesser than the node’s key.
* The right subtree of a node contains only nodes with keys greater than the node’s key.
* The left and right subtree each must also be a binary search tree.

Implement a tree type class named **BinarySearchTree** having following functionalities:

* **void insertNode(Type)** -> inserts a new node with a key of type Type in the Tree such that the property of BST is maintained.
* **bool search(Type)** -> returns true if key of type Type is found in the binary search tree and false if the key isn’t present.

Your program should be menu based and should ask the user what action he wishes to perform.

**Note:** Write constructor and destructor as well.

## Task 2

Diagram

Description automatically generatedConvert the **insertNode** function into a recursive code of insertion in the BST. Using your code, create the binary tree (given below).

**Note:** Mustsubmit proof of dry-run with your codes as well.

## Task 3

Now write three display functions through recursion which prints the tree in task3 in following orders:

1. Pre-Order Traversal.
2. In-Order Traversal.
3. Post-Order Traversal.

**Note:** Mustsubmit proof of dry-run with your codes as well.

## Task 4

Add a **deleteNode(Type)** function in your **BinarySearchTree** class which removes a node with a key of type Type from the Tree such that the property of BST is maintained and returns true if a node is deleted and false if not.

## Task 5

**Find the Minimum and Maximum Value in a BST**

**Problem**:

Write a function to find the minimum and maximum value in a Binary Search Tree (BST).

* The minimum value in a BST is the leftmost node.
* The maximum value in a BST is the rightmost node.

**Input:**

* A Binary Search Tree (BST) represented by its root node.

o The root node is a tree structure with the following properties:

* + Each node has a value, a left child, and a right child.
  + The left child of a node contains a value smaller than the parent node.
  + The right child of a node contains a value larger than the parent node.

**Output:**

* Two values: the minimum and maximum values in the BST.
* If the tree is empty, return None.

**Example:**

**Input**:

15

/ \

10 20

/ \ / \

5 12 18 25 **Output**:

Minimum: 5

Maximum: 25

## Task 6

**Find the Lowest Common Ancestor (LCA) in a BST**

**Problem**:

Given two nodes in a Binary Search Tree, find their lowest common ancestor (LCA).

* The LCA of two nodes n1 and n2 is the deepest node that is an ancestor of both n1 and n2.
* In a Binary Search Tree, the LCA is the node where:
  + One of the nodes lies in the left subtree, and the other lies in the right subtree, or o Both nodes are in the same subtree and the current node is an ancestor of both.

**Input:**

* A Binary Search Tree (BST) represented by its root node.
* Two integers n1 and n2 representing the values of the two nodes in the tree.
  + The tree contains unique values for each node.

**Output:**

* The value of the Lowest Common Ancestor (LCA) of the two nodes n1 and n2.
* If either n1 or n2 is not present in the tree, return None.

**Example:**

**Input**:

15

/ \

10 20

/ \ / \

5 12 18 25

n1 = 5, n2 = 12

**Output**:

LCA: 10 **Input**:

15

/ \

10 20

/ \ / \

5 12 18 25 n1 = 5, n2 = 25

**Output**:

LCA: 15