Data Structures and Algorithms

Lab Journal - Lab 12

Name: _____

Enrollment #: _____

Class/Section: _____

Objective

This lab session is aimed at introducing students to the 'Tree' data structure.

Task 1 :

Give answers to the following.

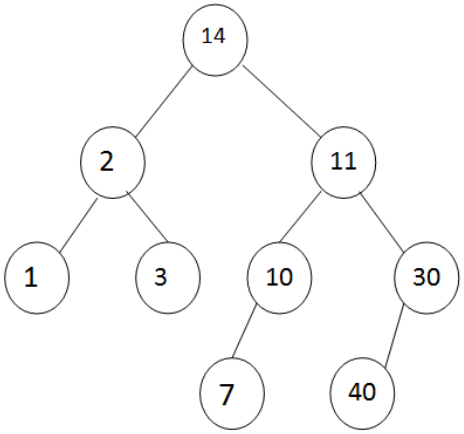
1.	For a complete binary tree of depth '*d*', find the following. a. Number of nodes at level *k* of the tree:_____ b. Number of leaves in the tree: _____ c. Total number of nodes in the tree: _____
2.	A complete binary tree has a total of 15 nodes. What is the depth of the binary tree? What would be the depth of a complete binary tree with '*K*' nodes?
3.	For the given binary tree, state the following. a. Number of leaf nodes: _____ b. Number of descendants of node containing 11:_____ c. Depth of the tree: _____ d. Parent node of the node containing 30: _____

e. Type of Binary Tree:_____

f. Level/Depth of node containing 10:_____

g. Children of the root: _____

h. Name the ancestors of node containing 7:_____



4. Traverse the binary tree given above in pre, post and inorder.

a. Preorder Traversal: _____

b. Post Traversal: _____

c. In-order Traversal: _____

5. Draw the expression tree of the given algebraic expression and traverse the tree in pre, post and inorder.

(a+b*c)+((d*e+f)*g)

Task 2 :

Implement the following exercises.

Exercise 1

Complete the given class to implement a binary search tree.

```
class Node
{
public:
      Node *left;
      Node *right;
      int data;
} ;

class bst
{
      Node *root;
      public:
      bst();
      bool isempty();
      void insert(int item);
      bool search(int item);
};
```

Exercise 2

Extend the above « BST » class to include functions for pre, post and in-order traversal of the tree. Use recursion to implement the traversal functions.

Exercise 3

Write program that creates a binary search tree using the BST class developed in Exercise. Create a menu and perform the following operations on user inputs. (Make a separate function for each).

a. Insert a node in the binary tree.

b. Count all leaf nodes of the tree*.

c. Count all non-leaf nodes of the tree.

d. Determines the size of a binary tree by counting the number of nodes in the tre e.

*Hints:
Use the following recursive definition: `getLeafCount(Node *)`

1. If node is NULL then return 0.
2. Else If left and right child nodes are NULL return 1.
3. Else recursively calculate leaf count of the tree using below formula.

Leaf count of a tree = Leaf count of left subtree + Leaf count of right subtree

Implement the given exercises and get them checked by your instructor. If you are unable to complete the tasks in the lab session, deposit this journal alongwith your programs (printed or handwritten) before the start of the next lab session.

S No.	Exercise	Checked By:
1.	Exercise 1	
2.	Exercise 2	
3.	Exercise 3	

++++++++++++++++++++++++++