Data Structures and Algorithms

Lab Journal - Lab 13

Name: ___________________________________

Enrollment #: ___________________________________

Class/Section: ___________________________________

Objective

This is the second lab session on the 'Tree' data structure and aims to enhance the understanding of operations on (binary search) trees as well as generation of expression trees.

Task 1 :

Give answers to the following.

1.	Given the following two traversals, attempt to construct the binary tree. Hints: In preorder traversal, first node is the root (while in postorder traversal the last node is the root node). Once root is identified, all nodes in the left and right sub-trees of root can be identified (repeatedly). Inorder: D B H E A I F J C G Preorder: A B D E H C F I J G

2. State what the following function is intended to compute.

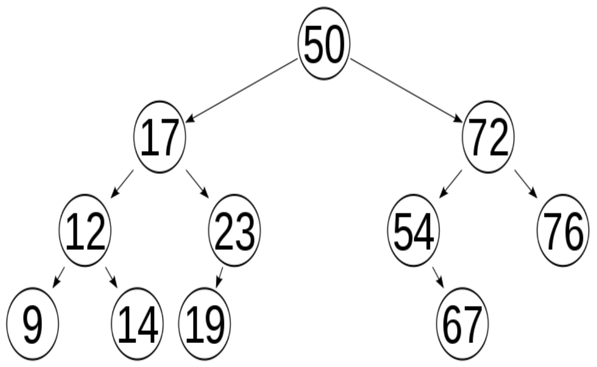
```
int A_Recursive_Counter(TNode *root)
{
 if(root==NULL)
      return 0;
 else

  return 1 + A_Recursive_Counter(root->left) + A_Recursive_Counter(root->right);
 }
```

3. What is the drawback if a sorted list of integers is used to generate a BST ?

4. A binary tree has a height of 5. What is the minimum number of nodes it can have?

5. Re-draw the given BST after deleting the node containing the value 17.



Task 2 :

Implement the following exercises.

Exercise 1

An effective way to represent Mathematical expressions is using the binary trees. The following algorithm can be employed to generate an expression tree from a given postfix expression.

```
1. Scan the postfix expression from left to right.
2. Create a node Curr
3. Get a symbol E from the expression.
4. If the symbol is an operand
        Set this operand as data member of the node Curr
        Push the node on the stack
5. If the symbol is an operator
        T2 = Pop()
        T1 = Pop()
        Attach T1 to the left and T2 to the right of Curr
        Set the operator as data member of the node Curr
        Push Curr (with child nodes attached) onto the stack
6. Repeat Steps 1-5 till end of expression
7. Pop the (only remaining) node from the stack which is a pointer to the root of the
   expression tree.
```

Consider the following class to model a node of the tree

```
class TNode{
public:
        char data;
        TNode * left;
        TNode * right;

};
```

Implement the function `TNode * constructTree(string postfix)` to convert a given postfix expression into an expression tree. Use the stack class in the STL library and create a Stack as follows.

```
stack <TNode *> s;
```

Call the function constructTree() with the postfix expression: "ab+ef*g*-"

Traverse the produced tree in postorder to verify its correctness. You should get back the input string.

Exercise 2

Write program that creates a binary search tree using the BST class developed in the previous lab session. Write functions to implement the following.

1. Find the minimum value in the BST
2. Find the maximum value in the BST
3. Given an integer value, find the address of its parent node.

Exercise 3

Create a class Student to store the name and ID of each student. Provide appropriate constructors, get(), set() and display() methods in the class.

Prompt user to enter data for 5 students and store the entered data in a BST. Each node of the BST should contain an object of class Student. Data to be arranged with respect to student names.

Once the data is entered, traverse the BST so as to display the sorted list of students (sorted on names). In addition, provide appropriate functionality to allow users search data for a given student using name as the search parameter.

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	

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