Birla Institute of Technology and Science, Pilani Hyd Campus
CS F211: Data Structures and Algorithms
2nd Semester 2024-25 Lab No: 9
Binary Tree, Expression Tree and Euler Tour

General Instructions

- You should use STL only for the program where it is mentioned explicitly. For other programs you should solve the given task without it.
- In addition to the methods already specified in the given code, you may also come up with your own methods to execute the tasks.

Program 1: (Prog1.cpp and linked_binary_tree.h attached)

This is an easier problem to Problem: P-7.1 in the Goodrich, Tamassia text. Given a Binary Tree rooted at *root, your goal is to perform a re-root operation on this binary tree. Meaning, given another pointer to a node in the tree (*newRoot), you should make this as the new root node of the binary tree using fix() and append() methods. The fix() function Extracts the subtree rooted at [newRoot] node from the tree rooted at [root] and append() method appends the previous tree (*root) as the right-most-child of the sub-tree rooted at *newRoot. You should get the output as shown below:

Output:

```
Initial Tree:-
Inorder: 3 7 5 12 11 14 9 15
Postorder: 3 7 12 14 11 15 9 5

Enter the new root of the tree: 12

After Re-Root operation:-
Inorder: 12 3 7 5 11 14 9 15
Postorder: 3 7 14 11 15 9 5 12
```

Task: Run it with a different test case and draw the binary tree on your notebook.

Program 2: (Prog2.cpp attached)

This is Problem: P-7.10 in the Goodrich, Tamassia text. In the lecture slides you were taught how to build a binary tree using a vector representation of a binary tree. It is possible to convert a *fully parenthesized valid expression* string directly to an *expression tree*. This experiment will further enhance your command over *Recursion*. Here are the functions that are used in the code:

- buildExpressionTree(): A method to build an expression tree directly from the given expression string with operators as internal nodes and operands as leaves.
- evaluateExpression(): A method to evaluate the expression represented by the expression tree. It utilizes the eval() method wherever needed.

The output should be as shown in the next page:

Output:

```
Expr Tree 1 = ((5+5)*(3-1))
Expr Tree 2 = (5+((3-4)*2))
Expr Tree 3 = (8*((6+(3-(4/(3/2))))-1))
Evaluating...
Expr Tree 1 Result = 20
Expr Tree 2 Result = 3
Expr Tree 3 Result = 42.6667


...Program finished with exit code 0
Press ENTER to exit console.
```

Task: Run it with a different test case and draw atleast one expression tree on your notebook.

Program 3: In the lecture class, you were taught how we can find LCA (least common ancestors) and Dependents in a Binary Tree Using Euler's Tour. In this task, given a binary tree you need to:

1. Find the Lowest Common Ancestor (LCA) of two given nodes.
2. Determine the dependents (number of nodes in the subtree rooted at a given node).

You must use Euler's Tour to solve this problem efficiently.

Input:

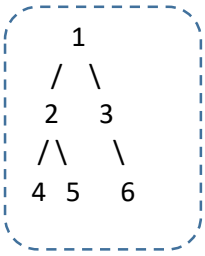
- A binary tree represented using a linked structure or a vector implementation as implemented in previous problems.
- Two nodes, u and v, for which the LCA needs to be found.
- A node x, for which the number of dependents (subtree size) needs to be calculated.

Output:

1. The LCA of nodes u and v.
2. The number of dependents (subtree size) of node x.

Program3.cpp is given with part code (attached). Complete it by writing the missing functions.

An example is as given below:



❯ LCA of 4 and 5 is 2.
Dependents of 2 is 3.

- What is LCA of 3 and 6? Find out the output from your C++ code.

Also, check do you get the LCA of 4 and 6 as 1 or not. Take few different trees and compute these values for few more cases.
