Lab 6: Converting Recursion to Loop

1. Introduction

In lab4, we re-implemented some methods of a LinkedList class using recursion, which were originally implemented using loops. In this lab, we will consider the opposite direction: to convert from recursion to loop. More specifically, you will first implement three tree traversal functions (pre-order, in-order, and post-order traversal) using recursion, and then with loops. To achieve this, you will emulate recursive function calls by using your Stack class - this will represent the function call stack - and your State class, which captures the state of program execution at a particular moment within each (recursive) function call. We will discuss this more during lab time. The goal of this lab is for you to discover how recursive (and general) function calls are carried out in program execution.

The code skeleton is under: **/comp/15/files/l6**

1. Log in to the CS homework server.
2. Move to your **comp15** directory that you created in Lab 1.
3. Create a directory named **lab6** under the comp15 directory.
4. Move to the lab directory.
5. Copy the code skeleton to the current working directory.
6. Leverage the features of Git to manage your progress toward writing the program.

The code skeleton provides you with 11 files: **test.cpp**, **Node.hpp**, **Node.cpp**, **LinkedList.hpp**, **LinkedList.cpp**, **Stack.hpp**, **Stack.cpp**, **BTNode.hpp**, **BTNode.cpp**, **State.hpp**, and **State.cpp**. (BTNode class is to represent binary tree nodes.)

- First, take a look at the files to find the relationships among the BTNode, State, Stack, LinkedList, and Node classes.
- The BTNode, State and Node classes have already been implemented for you. If necessary, you can implement the default constructor, copy constructor, assignment operator and destructor.
- Note that the LinkedList class supports only selected operations, and the copy constructor and assignment operator are commented out so that this lab can be completed within our lab time. However, you are encouraged to work on them later on. (Same for the Stack class).
- **test.cpp** provides you with the skeleton of the six functions that you will implement (See Requirements section) and several tests with assert() that your implementation is supposed to pass.

2. Requirements

Recommendation: Take a backup of your lab5 before proceeding Step 1 and 2 below.

1. Modify the LinkedList class that you implemented in lab5, so it can hold `State*`, instead of `char`.
    a. Check the given **LinkedList.hpp** and pay attention to the type signatures of the at() and addToFront() methods.
    b. (Node class has already been updated for you.)
2. Modify the Stack class that you implemented in lab5, so it can hold `State*`, instead of `char`.
    a. Check the given **Stack.hpp** and pay attention to the type signatures of the push() and top() methods.
3. Check the main() function in **test.cpp**, and then using a pen and a sheet of paper:
    a. Sketch how the 10 binary-tree-nodes are connected.
    b. Simulate the pre-order, in-order, and post-order traversal on the binary-tree-nodes, in which the "six" BTNode is the root.
    c. Write the expected traversal orders.
4. Implement the following three functions in **test.cpp** using recursion:
    a. preOrder(), inOrder(), postOrder()
5. Implement the following three functions in **test.cpp** using loop:
    a. preOrderByLoop(), inOrderByLoop(), postOrderByLoop()
6. Pass the tests with assert() written in **test.cpp** with no memory leaks and no memory errors.

3. README

Create the README file that includes the following categories with appropriate section headers.

1. **Name**: Your name.
2. **Date**: The last updated date.
3. **Summary**: A brief summary of the lab.
4. **Files**: A list of files that are necessary to build and test the program.
5. **Instructions**: A sequence of commands to compile and test the program. Note that you are expected to report procedures without using the make command.
6. **References**: A list of citations to information used to complete the lab.

4. Submission

Submit your files listed below using Gradescope.

Files: **test.cpp BTNode.cpp BTNode.hpp State.cpp State.hpp Stack.cpp Stack.hpp LinkedList.cpp LinkedList.hpp Node.cpp Node.hpp README**