Approach and Screenshots

## Goal:

Implement a Double-Threaded Binary Tree and add in-order and reverse-order printing without resorting to recursion — Project 5.2 in the text.

## Reference Material:

Data Structures and Algorithm Analysis, page 200.

## Approach:

11/14/2023

I started this project by reading the project description for project 5.2 in the text to understand where our project’s goal came from. Then I read each file that was provided starting with BinNode.h, then BSTNode.h, and finally BST.h. Reading in this order allowed me to understand the already structure inheritance of this project.

I started this project by creating a test file that allows me to run my code following the constraints that are required per this assignment. This program, BSTtest.cpp will build the BST, assign the appropriate values, and demonstrate that the Binary Search tree has double-threaded functionality.

After attempting to create a singular test file for this project, I will instead create the test file as I go, starting by adding the first few required items for the project, and testing them as I move forward.

11/15/2023

For today, I received feedback from the first project, so I spent my time before my shift ensuring that I was able to use the makefile system for my project.

11/16/2023

I started on the modification requirements in the instructions by comparing if a value was entered for a node’s left and right child, and if so, we know that the node’s children are not threads.

Something that I’ve found conceptually difficult so far in this project ensuring that the modifications to ensure a threading functionality do not impede the ability of the function originally.

11/17/2023

I was getting frustrated with my lack of progress after this point, so I decided to restart by completely removing all changes made to the files and then I started by creating the print, printinorder, and printreverse functions before making the BST double threaded.

11/18/2023

This morning I was able to create a fully functional BST that did not rely on recursion to print. The key factor that was preventing me from being able to do this before was the classification of nodes in inserthelp(). Specifically, I was comparing if the root was NULL or if the root was threaded in the same line, but moving these to different locations in the method allowed me to properly construct my BST.

A screen shot of a computer

Description automatically generated