**Tammy Hartline**

**03/31/2023**

**CS-300 Week 5**

**Binary Tree Search Assignment**

1st FIXME BinarySearchTree()

SET root EQUAL to null pointer

2nd FIXME InOrder()

CALL THIS recursive inOrder function and pass in root variable

3rd FIXME PostOrder()

CALL THIS recursive postOrder function and pass in root variable

4th FIXME PreOrder()

CALL THIS recursive preOrder function and pass in root variable

5th FIXME Insert(Bid bid)

IF root is EQUAL to null pointer

THEN root is empty

IF root IS empty

CREATE NEW Node bid and SET that EQUAL to root

ELSE

CALL addNode recursive function for this bid AND

PASS in (ADD) bid to root

6th FIXME Remove(string bidId)

COMPLETE recursive function called removeNode(Node\* node, string bidId) START

Node\* BinarySearchTreee::removeNode(Node\* node, string bidId)

IF node is EQUAL to null pointer

RETURN empty node;

IF bid id, WHEN compared to this bid id is LESS than 0

SET left node EQUAL to CALL recursive removeNode(left node &bid id)

ELSE IF bid id, when compared to this bid is GREATER than 0

SET right node EQUAL to CALL recursive removeNode(right node and bid id)

ELSE

NO more nodes have children, therefore

IF left node IS EQUAL to null pointer AND right node IS EQUAL to null pointer

THEN DELETE node

SET node EQUAL to null pointer

ELSE IF left node is NOT EQUAL to null pointer & right node EQUALS null pointer

THEN SET NEW node (temp node) EQUAL to node

SET node EQUAL to left node

DELETE temp node

ELSE IF right node IS NOT EQUAL to null pointer & left node EQUALS null pointer

THEN SET NEW node (temp node) EQUAL to node

SET node EQUAL to right node

DELETE temp node

ELSE (START OF ELSE)

SET temp node EQUAL to right node

START WHILE LOOP

WHILE the temp left node IS EQUAL to null pointer

SET the temp node EQUAL to left temp node

END WHILE LOOP

SET node bid EQUAL to temp bid

SET node right EQUAL to CALL recursive removeNode(right node and bid id)

END ELSE

END OF removeNode COMPLETION

IN non-recursive Remove(string bidId)

CALL recursive removeNode(bid id from root)

7th FIXME Search(string bidId)

SET current node EQUAL to root node

WHILE current node IS NOT EQUAL to null pointer

IF this bid id WHEN compared to current bid id EQUALS 0

RETURN current bid

ELSE

SET current EQUAL to right

8th FIXME recursive function addNode(Node\* node, Bid bid)

IF nodes bid id WHEN compared to bid id is GREATER than 0

IF left node EQUALS null pointer

SET left node EQUAL to new Node bid

ELSE

CALL addNode to bid for left node

ELSE

IF right node EQUALS null pointer

SET right node EQUAL to new Node bid

ELSE

CALL recursive addNode to bid for right node

9th FIXME recursive function inOrder(Node\* node)

IF node IS NOT EQUAL to null pointer

CALL inOrder and pass through left node

OUTPUT bid id, title, amount and fund

CALL inOrder and pass through right node

10th FIXME recursive postOrder(Node\* node)

IF node IS NOT EQUAL to null pointer

CALL postOrder and pass in left node

CALL postOrder and pass in right node

OUTPUT bid id, title, amount and fund

11th FIXME recursive preOrder(Node\* node)

IF node IS NOT EQUAL to null pointer

OUTPUT bid id, title, amount and fund

CALL preOrder and pass in left node

CALL preOrder and pass in right node

**CODE REFLECTION**

This week’s assignment purpose was to give us a better understanding of the usage of binary trees and their relation to programming. The programs’ purpose is to present users with menu options that allow them to load a bid file, view the information in that file, search for a specific bid, and/or remove a bid. While I still feel like I am struggling to keep nodes, and the recursive operations organized in my mind, with their purposes, after watching each lecture in the update section several times, I do feel a little more confident when developing both the pseudocode and programming code. I by far, had an easier time with this lesson, versus those from the previous weeks. This may be due to practice and resubmissions, but either way, good to know that I am, at least on some level, retaining and beginning to understand the material for this class.

The main issue I faced this week was honestly, once again with the pseudocode. I have a hard time not translating the pseudocode into actual programming code, or I tend to make it to long, and detailed, and have found I am still working to find the right balance between the two. My other issue this week, was the first version of this assignment, when complete, and user selected option 2, instead of the entire file printing, only 3 lines printed. This seemed to continuously happen, regardless of what I changed. My solution to this was to delete the entire project and start fresh. Upon doing this, I was able to finish the program and select option 2, resulting in the entire file printing. The only bad thing about my method of deleting everything and starting from scratch, is that I now have no idea what the problem is, or how the changes made in the second version fixed the issue. So, moving forward, when I run into an issue with similar circumstances, I plan to save my previous version with testing tag in the folder name, so I can compare the two later on, to determine what the actual problem was.