Philip Trinh

SNHU/CS-300

3/31/2023

**Reflection and Pseudocode module 5 Binary Tree**

In this assignment, I am working with information from a municipal government data feed containing bids submitted for auction of property. I was provided an incomplete BinarySearchTree.cpp program in which I need to make changes to BinarySearchTree class, insert, remove, search methods. To tackle this project, I reviewed the ZYBook lessons on chapter 6 which contain examples for creating a Binary Search Tree, insert, remove, and search methods. The Binary Search Tree important is to initialize variable, make sure to create tranverse tree in many order to include InOrder, PostOrder, and PreOrder methods. Insert method will add new bid into the tree to be search. Search method will search the tree for the specified bid by comparing the bidId which will significantly save time, if greater will search right side branch and lesser will search left side. Finally the remove method will remove the specified bid. I enjoyed following the instructor provided in the video lecture and read the comment instruction provided from BinarySearchTree. I ran into problem when I try to use g++ to compile the code, it show “int \_\_cdecl remove(const char \*\_Filename)"and reference “str.erase(remove(str.begin(), str.end(), ch), str.end());” within the code. I was able to fix it by #include <algorithm>. Afterward the code compiled the code without problem. Everything tasted like it was working as intended.

**Pseudocode**

**Create BinarySearchTree, Add, Remove, Search**

Declare class BinarySearchTree and define structure to hold bids, struct Node, private and public methods for addNode, inOrder, preOrder, postOrder, and removeNode.

Constructor: initialize root variables to nullptr

Destructor: recurse from root deleting every node

Traverse tree in order

Insert method:

If root is empty then add new node to root

Else implementing inserting a bid into the tree

If node is larger then add to the left side of the tree

If no left node, new node become the left node

Return

Else

Recurse down to the left node

Else if node is lesser then add to the right side of the tree

If no right node, new node become right node

Return

Else

Recurse down the right node

Binary Search Tree order:

inOrder root:

if node is not empty

in order node to the left

then print output

in order node to the right

postOrder root:

if node is not empty

postOrder node to the left

postOrder node to the right

then print output

preOrder root:

if node is not empty

print output first

preOrder node to the left

preOrder node to the right

Search Tree method:

Set current node equal to root

While current node is not empty

Keep looping downwards until reach bottom or a matching bidId found

Return current bid if match found

Else if bid is smaller than current node then traverse left

Else larger then traverse right

Return bid

Remove method:

If node is empty

Return node since nothing to remove

Else if compare the bid is less than bidId

Node point to the left and remove

Else if compare the bid is greater than bidId

Node point to the right and remove

Else

If match is found, only leaf node

Delete the node

Node is nullptr

Else if left side have one child

Create temporary node and point node to left node

Then delete temp node

Else if right side have one child

Create temporary node and point node to right node

Then delete temp node

Else if there are two children

Find minimum of the right subtree nodes for BST

Create temporary node point node to the right

While left node is not empty

temporary node recursively point to left, because the smaller node should be on left side

when can't go down anymore, found the minimum bid, copy it to the temporary node to replace

point to node on the right side, remove the node on the right with the temp bidId, because tempbidId is the minimum node

return node