5-2 Assignment: Binary Search Tree

Fred Wahab

CS-300

**Code Reflection**

Like last week, this week’s assignment had us implementing a singly linked list to hold a collection of bids loaded form a .csv file. A binary search tree is defined as a data structure that consists of the node-based binary tree where each node consists of at most 2 nodes that are referred to as child nodes. The purpose is to store data in an organized way.

This week’s starter code was exceptionally difficult to decipher. The code starts out broken and there appears to be declarations missing that are not flagged to declare or fix. This makes a difficult assignment near impossible.

**Pseudocode**

**FIXME (1): Initialize housekeeping variables**

BinarySearchTree::BinarySearchTree() {

root is equal to nullptr

}

**FIXME (2): In order root**

void BinarySearchTree::InOrder() {

call inOrder fuction and pass root

}

**FIXME (3): Post order root**

void BinarySearchTree::PostOrder() {

postOrder root

}

**FIXME (4): Pre order root**

void BinarySearchTree::PreOrder() {

preOrder root

}

**FIXME (5): Implement inserting bid into tree**

void BinarySearchTree::Insert(Bid bid) {

if (root equal to null ptr) {

root is equal to new node bid

}

else {

add Node root and bid

}

}

**FIXME (6): Implement removing a bid from a tree**

void BinarySearchTree::Remove(string bidId) {

remove node root bidID

}

**FIXME (7): Implement searching the tree for a bid**

Bid BinarySearchTree::Search(string bidId) {

set current node equal to root

while current node does not equal nullptr {

if match found, return current bid

if bid is smaller than current node then traverse left

else larger so traverse right

}

Bid bid;

return bid;

}

**FIXME (8): Implement inserting a bid into the tree**

void BinarySearchTree::addNode(Node\* node, Bid bid) {

if node is larger then add to left

if no left node, this node becomes left

else, recurse down the left node

else {

if no right node, this node becomes right

else, recurse down the left node

}

}

**FIXME (9): Pre order root**

void BinarySearchTree::inOrder(Node\* node) {

if node is not equal to null ptr {

InOrder not left

output bidID, title, amount, fund

InOder right

}

}

**FIXME (10): Pre order root**

void BinarySearchTree::postOrder(Node\* node) {

if node is not equal to null ptr {

postOrder left

postOrder right

output bidID, title, amount, fund

}

}

**FIXME (11): Pre order root**

void BinarySearchTree::preOrder(Node\* node) {

if node is not equal to null ptr {

output bidID, title, amount, fund

postOrder left

}

}