Nicholas Ang
CS302

Homework 3 Documentation

Binary Node Class Implementation:

I implemented this class based on the header file given in the slides. There are three constructors that create a binary node based on the values given. The default constructor sets everything to null values while the parameterized constructor gets an item value. The parameterized constructor can be set with children pointers or set to nullptr. The setters and getters get the targeted pointer or item value. The getItem function gets the item value inside the node and the setItem function sets the node with a item value. The getRightChildPtr and getLeftChildPtr return either the shared pointer or normal pointer of the child pointers. The setRightChildPtr and setLeftChildPtr set the child pointers of the node. The isLeaf function checks whether the node has no child pointers to see if it is a leaf.

Binary Node Tree Class Implementation:

I implemented this class using the header file in the slides but slightly modified some of the functions to accommodate c++14 and the type auto. This class has 11 protected methods, 17 public methods and an operator overload for =. The slides give the implementation of copyTree, destroyTree, getHeightHelper, add, balancedAdd, and the constructors/destructors. The getNumberOfNodesHelper is similar to getHeightHelper but adds 1 for each node in the tree by recursively calling itself. The removeValue was modified to return a shared pointer since auto types cannot be used with virtual. The function searches through the tree until the target is found and then calls moveValuesUpTree to move the node to the bottom where it can be removed from the tree. The moveValuesUpTree replaces the nodes and shifts the values up and moves the target to the bottom. The findNode function is similar to the removeValue function but does not remove the node from the tree. It was also modified to not be an auto type and return a shared pointer. The preorder method starts from the root and calls the visit function to output the item value before recursively calling preorder on the left and right pointers. The inorder method recursively calls itself for the left, then calls visit, and then recursively calls itself for the right. The postorder method recursively calls itself from the left and the right then visits to output the value. The isEmpty function checks if the rootptr is nullptr. The getHeight and getNumberOfNodes call their respective helper functions. The getRootData function gets the item value inside the rootptr. The clear function calls destroyTree and resets the shared pointer to delete the item. The getEntry function calls the findNode function to find the targeted node and calls getItem to get the value inside. The contains function calls the findNode method and returns true or false depending if the node is found. The operator overload clears the tree before checking for self assignment. Then it calls copyTree to copy the values over to the tree.

Binary Search Tree Class Implementation:

I implemented this class using the header file in the assignment but slightly modified some of the functions to accommodate c++14 and the type auto. There are 5 protected methods, 4 constructors/destructors, 13 public methods, and an operator overload for =. The five protected methods are given in the slides. The default constructor creates a binary search tree with nullptr as the rootptr. The parameterized constructor creates a binary search tree with a rootptr that points to a node with a given item value and nullptr children. The copy constructor calls the inherited copyTree function that it got from the BinaryNodeTree class to copy the values to a new tree. The destructor calls the inherited destroyTree function to destroy the tree. The isEmpty function is the same as the BinaryNodeTree implementation of it. The getRootData function gets the item value inside the node of the rootptr and outputs an error message if the tree is empty. The setRootData sets the node at rootptr to the given item value. The add function is the same as the BinaryNodeTree implementation of it but uses the placeNode function. The remove function is the same as the BinaryNodeTree implementation but uses a modified removeValue function to account for every case. The clear function calls the inherited destroyTree function and resets the shared pointer to delete the root. The getEntry and setEntry functions are the same as the BinaryNodeTree implementation. The preorderTraverse, inorderTraverse, and postorderTraverse methods call their respective inherited traversal methods to traverse the tree. The preorderTraverse starts from the root. The inorderTraverse should output a sorted list. The postorderTraverse ends the output of the list with the root. The operator overload for = is the same as the implementation in BinaryNodeTree.

Precond Violated Except Class implementation:

This class outputs a string error message when the constructor is called.

Not Found Exception Class Implementation:

This class outputs a string error message when the constructor is called.

Main CPP file:

The main cpp file outputs a menu with 5 options. The first option creates a binary search tree with 100 unique random values. It also outputs the height of the tree and the values in the tree in the order that they were created. The second option traverses the tree using preorder traversal. This calls the preorderTraverse method and outputs starting from the root. The third option traverses the tree using the inorder traversal method. This calls the inorderTraverse method and outputs the smallest value to the biggest value due to being a binary search tree. The fourth option traverses the tree using the postorder traversal method. This calls the postorderTraverse method and outputs the values from the left and right, ending at the root. The fifth option exits the program when selected. For a binary search tree, the height for 100 unique random values from 1-200 is around 12 to 13.

I compiled the treeMain executable using cmake. In the folder with the CMakeLists.txt file, I type cmake ./ into the terminal. Then I entered make all to compile my code and create the executable treeMain. The numbers have to be generated using option 1 before using the traversal methods which are options 2, 3, and 4 for the menu.