COSC 2030 Computer Science II
Spring 2018

Lab 8: Binary Tree Traversals and Statistics

Monday, April 2 2018

Background:

In class we have begun to study Binary Trees. In this lab we will write functions that yield some important information about a binary tree, such as the size of the tree, how many leaves are in the tree, and what is stored in the leftmost node in the tree.

Tree Vocabulary:

size	the number of nodes in the tree
height	the number of nodes on the longest branch in the tree
leaf	a node whose left and right subtrees are empty trees
leftmost	the first node visited in the inorder traversal of the tree

Download the following files: BinaryTreeDriver.cpp, BinaryTree.h, and RandomUtilities.h.

Create a new project, add the files, and run the program. Enter help, or any other unknown request, to see a list of known requests.
The request "build <size>" means you should enter the string "build", a space, and then the size of the tree you want built, a number.

NOTE: The trees are displayed with the left side of the tree at the top of the display; the right side is at the bottom.

Lab Assignment:

Experiment with the current BinaryTree code. Become familiar with the code input and output.

Modify BinaryTree.h by providing a correct implementation of the size helper method. The public size method returns the size of this BinaryTree. The private size helper method returns **the size of the binary subtree** whose root is referenced by subtree. **L8P1**

Modify BinaryTree.h by providing a correct implementation of the height helper method. The height method returns **the height** of this BinaryTree. **L8P2**

Modify BinaryTree.h by providing a correct implementation of the leaves helper method. The leaves method returns **the number of nodes** whose left and right subtrees are empty trees in this BinaryTree. **L8P3**

COSC 2030 Computer Science II
Spring 2018

Modify BinaryTree.h by providing a correct implementation of the leftmost helper method. The leftmost method returns the **value that is stored in** the leftmost node, not the index or the address of that node. **L8P4**

Modify BinaryTree.h to add code to do a post-order traversal for this BinaryTree. At each node the post-order traversal will first print the entries in the subtree on the left-hand side, then print the entries in the subtree on the right-hand side, and last print the entry in the root node. Model your code after the pre-order traversal implementation. **L8P5**

Modify BinaryTreeDriver.cpp to add a request for a postorder traversal to the user's menu. **L8P6**

Turn In:

Upload the modified BinaryTree.h and BinaryTreeDriver.cpp files (with your name in comments) to Github.

Problem	Points Possible
L8P1	3 pts
L8P2	2 pts
L8P3	5 pts
L8P4	5 pts
L8P5	3 pts
L8P6	2 pts

Total Points Possible: 20
Due: Sunday April 8@ 11:59PM