**CMSC 3621**

**Lab 3**

Implement two additional methods (and a helper function) for “Binary search tree” as follows:

1)

**template** <**class** Entry>

**int** Binary\_tree<Entry>::leaf\_count() **const**

/\*

Post: The number of leaves in the binary tree is returned.

Uses: The function recursive\_leaf\_count

\*/

2)

**template** <**class** Entry>

**void** Binary\_tree<Entry>::nonrecursive\_postorder(**void** (\*visit)(Entry &))

/\*

Post: The tree has been traversed in postorder sequence. The implementation cannot use recursion.

\*/

Requirements:

1. Your implementation must be based on "Binary search tree" under Assignment 3 in the URL below:

<https://cs2.uco.edu/~gqian/cmsc3621/assignments>

1. You are required to implement leaf\_count and recursive\_leaf\_count in **Binary\_tree.h**. recursive\_leaf\_count should be a recursive helper function invoked by public method leaf\_count. Use size and recursive\_size as examples for your implementation. When you finish the two methods in Binary\_tree.h, uncomment line 89 in main.cpp to test-run your program. You can use option 1 in the user interface to import the provided data file (ins1.txt) and then use option l to display the number of leaves. For ins1.txt, the result should show that the binary search tree has 10 leaves.

Note: a *leaf* node is defined as a node with no children.

1. You are required to implement nonrecursive\_postorder in **Binary\_tree.h**. nonrecursive\_postorder should be a non-recursive public method. Use nonrecursive\_preorder and nonrecursive\_inorder as examples for your implementation. When you finish the method in Binary\_tree.h, uncomment line 77 in main.cpp to test-run your program. You can use option 1 in the user interface to import the provided data file (ins1.txt) and then use option 7 to display the tree nodes in postorder. You can compare your result with that of option 6, which uses a recursive implementation for postorder traversal.
2. Create a subdirectory named "lab3" under your server account on cs.uco.edu. Upload all your source code files, "makefile", and "ins1.txt" to your server account under "lab3". The instructor should be able to use "make" to compile and test-run your program.
3. Use nano or vi to create a file named "answers" under "lab3". Add your answers to the following questions into "answers".

Review nonrecursive\_preorder, nonrecursive\_inorder, and your implementation of nonrecursive\_postorder. In a few sentences, describe your understanding of the use of the two stacks (proc\_st and st) in the three methods. Your answer must cover the three questions below:

* What is the purpose of having the proc\_st stack in the implementation?
* How does proc\_st coordinate with the st stack? In other words, what are the relationships between the entries in the two stacks?
* How are the true and false values used in the proc\_st stack?

Submission:

The following files must be available under "lab3" in your server account on cs.uco.edu:

1. The source code files, "makefile", and "ins1.txt" for "Binary search tree" with your implementation of leaf\_count, recursive\_leaf\_count, and nonrecursive\_postorder.
2. The compiled, executable program "bst"
3. The file "answers" with your answers to the questions in step 5)