Binary Search Tree (BST)

Purpose

In this assignment you are going to create and do operations on a Binary Search Tree (BST).

binaryTree.h, binarySearchTree.cpp and BinarySearchTreeSol.cpp

Your program cannot use any existing C++ STL collections to implement this assignment. You are to use the provided files for developing the assignment. You are to instrument code into the provided files.

A BinarySearchTreeSol.zip file is provided that contains the .h and .cpp files that are required to use in the assignment. Do not modify or change any of the provided code. This includes the definitions and headers in the provided files.

You must implement all the member functions that are stubs in the required files. Some of the member stubs have dummy return values so that the stubs will compile. The stubs may have a return set to a reasonable default value (nullptr or false for example) so that they compile.
You must go through and instrument the stub code so that the correct operation is done, and any stub correct return values are performing correctly.

Areas in the classes that require code development are marked with //$.
For more information on the required operations for the stub methods and operators see the files and the comments that are in front of the stub function members and the associated prototypes.

main() Function

The main() function must be a unit test driver that shows that your implementations are working correctly.

You must show in the main unit test driver that all the required operations for all the paths through the code in the files are executed at least once by the main test driver.

You must display what capability is being tested and display the result of the BST operation to show that the capability worked. Incomplete or missing testing of paths and operations will result in point loss on the assignment.
Use int for the parameter type argument for the template parameter when creating objects to use in the main program.

Summary

This assignment is going to take a while to get working, so get started right away.

The incremental development cycle should proceed as follows:

Write the code for one-member function or operator.
Then in the main test driver add the code that thoroughly demonstrates that the member is working correctly.
Repeat this approach for each member development before you go to the next member to be developed.

The recommend order for developing is:

- Constructors (instantiate an object)
- operations on the BST

Continue to develop incrementally the BST and test in main() until you have the whole design working.

By doing this incremental development you will have a better idea which function is failing (because it is probably the code you just wrote). Also, make sure you test and fix a member function before you go onto the next one.