**Programming Threaded Binary Tree Assignment Instructions**

**Overview**

Implement a Double-Threaded Binary Tree and add in-order and reverse-order printing without resorting to recursion — Project 5.2 in the text.

**Instructions**

Using the following supplied C++ files implement a right and left threaded binary search tree. The files provided to you come from our text, with some minor modifications, and implement a BST based dictionary. You must modify this BST-based dictionary to implement a threaded BST. Specifically, you will need to make the following modifications:

* BSTNode.h
  + Add Boolean instance variables to indicate whether a node pointer is a thread or regular pointer. You may not add any additional pointers to BSTNode. The whole idea of threads is that they take advantage of unused BSTNode pointers and thereby reduce the binary tree’s wasted overhead.
  + Add setter/getter methods to access the Boolean instance variables or modify existing setters/getters as necessary.
* BST.h
  + Rewrite method inserthelp() to take advantage of the modified BSTNode in which a pointer can be either a regular pointer or a thread.
  + Modify method printhelp() to work with threaded nodes.
  + Add method printInorder() to do an inorder printing of your tree without the use of recursion.
  + Add printReverse() to do a reverse order printing of your tree without resorting to recursion.
  + You may add private helper methods as needed to make modifying or creating the methods above easier.
  + Note: I’ve commented out the destructor since the destructor relies on clearhelp() and clearhelp(), as currently written, won’t work with threaded trees.
  + Note: You do not have to implement any of the functions I’ve removed or deleted. Only the functions I’ve specified above, along with any private helper functions you need, must be implemented.
* Approach – in a Word document explain your implementation approach. Describe how you plan to accomplish this assignment. List where in your source files (source file names and method/function names) your implementation can be found. Use methods printhelp, printInorder, and printReverse to demonstrate your program in action. Include screen shots of your program’s output.
* Since the BST node takes a key value pair, insert the following <int, string> values (in the order provided) to build your tree.
  + 77, “seventy-seven”
  + 70, "seventy"
  + 75, "seventy-five"
  + 66, "sixty-six"
  + 79, "seventy-nine"
  + 68, "sixty-eight"
  + 67, "sixty-seven"
  + 69, "sixty-nine"
  + 90, "ninety"
  + 85, "eighty-five"
  + 83, "eighty-three"
  + 87, "eighty-seven"
  + 65, “sixty-five”

## Files provided

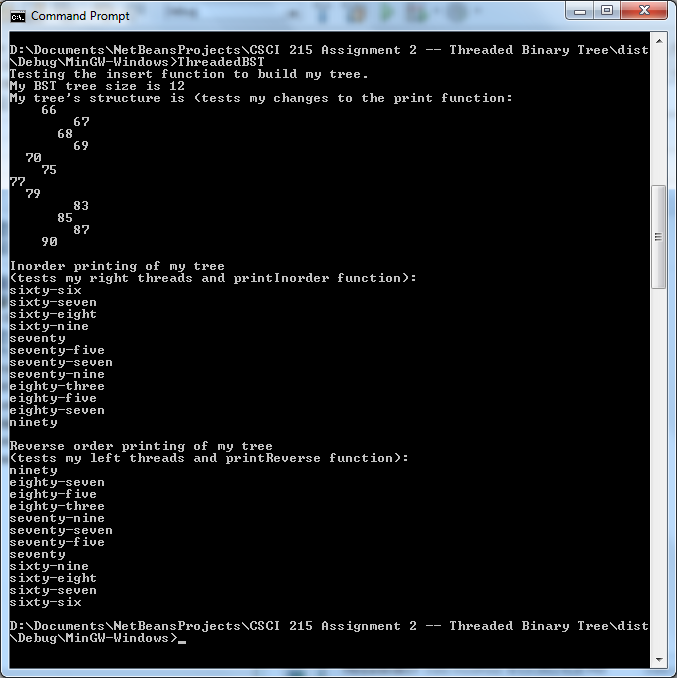
* BST.h
* BSTNode.h
* BinNode.h
* dictionary.h

## Assignment Submission

Put the following files into a zip file and submit your assignment to the assignment link in Canvas:

* Your Visual Studios entire project file. At the absolute minimum submit all your .cpp and .h files. If I cannot build your project with the files you give me, you will not receive any credit for your work.
* Word document describing your approach to solving this problem.
* Word document with screen shots (of your entire test run) with integrity statements. You only get credit for the functions you have demonstrated work. Be sure that the output in your screen shots is clear enough that I know exactly which functions are being tested, and that you explain the results in such a way that the test results are clear. This BST uses templates <Key, E>. For your test run use the following key/value pair type combination: <int, string>.

Here is an example of my test run showing both printhelp, printInorder, and printReverse:



Put all your files into a single zip file and submit it to Canvas.

## Tips

This is a deceptively challenging project. Do not wait until the last minute to start working on it or you will go crazy (and probably not succeed in finishing your project). I recommend that you start with your Word document and describe your implementation approach and then use that to guide your actual implementation.

Break the project into pieces and think about how you are going to accomplish each piece. Start by getting the BST files I’ve given you running before making any changes to them and be sure you understand how the author has implemented his BST. Then add your Boolean instance variables to the BSTNode.h file and implement your setter/getter methods for your threads.

Next start thinking about inserthelp. You pretty much have to gut the entire method and start from scratch. Think about node states and how each state affects the insert operation. For example the state of root when it is the only node in the tree is left and right child equal NULL. How will you add a node to root and implement your successor and predecessor pointers? Now look at the state of the second node. How is it different from the state root was in, and how does that change future insertions? For me it helps drawing the tree out as I build it, so that I can visualize what it is I am attempting to do.

## Debugging Your Code

A big part of this assignment is debugging your code, so do not expect your instructor to do this for you. Having completed the pre-requisite courses for this class, you should already have some experience finding coding errors and fixing them. This class will give you plenty of opportunities to further refine those skills and you can only do that by **wrestling with the problem**. Here are some debugging tips:

* Build a little, test a little. Don’t write all your code before you start debugging it. Break your work into logical chunks that can be compiled and debugged and build them one at a time. For this project, I would start by compiling and running the files I’ve given you before you make any changes. Then build your threaded tree a node at a time by modifying inserthelp and BinNode as necessary and use the debugger to determine whether or not your pointers and threads are being correctly applied.
* Learn to use the debugger if you haven’t already. The debugger allows you to step through your code one step and a time and see what happens in memory while you’re doing it. When students come to me with problems, I first try to isolate where the problem is logically based on what the program is doing and then I use the debugger to find the actual fault.
* Be willing to walk away from your computer and give your mind a rest. I find that the solution to a problem often comes to me while I am doing something else. I have never regretted stepping away from my computer to let the problem “percolate” in my mind, but I have often regretted not doing this. This is one of the reasons you should never wait till the last minute to start working on your program; by doing that you are not giving yourself the time to walk away.

Make sure you take full advantage of the debugger in your development environment. Debuggers are invaluable for examining the state of pointers and variables as they make their way through the program.