Programming Assignment 2  
Double-Threaded Binary Tree

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

Using the following supplied C++ files implement a right and left threaded binary search tree (see <https://algorithms.tutorialhorizon.com/double-threaded-binary-tree-complete-implementation/> for more information on doubly threaded BSTs). The files provided 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 bit or Boolean fields to indicate whether a node pointer is a thread or regular pointer (you get extra credit if you use a bit field). 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 binary tree wasted overhead.
  + Add setter/getter methods to access the bit or 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 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.
* Approach – in a Word document explain your implementation approach. Tell me how you accomplished this assignment. Tell me where in your source files (source file names and method/function names) I can look to see your implementation. Use methods printhelp, printInorder, and printReverse to demonstrate your program in action. Include either screen shots of your program’s output or an output file that your program writes to (you can use redirection to easily create your output files).
* Since the BST node takes a key value pair I want you to use 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”
* Smart Pointers – I **do not** recommend the use of smart pointers for this assignment.

## Files provided:

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

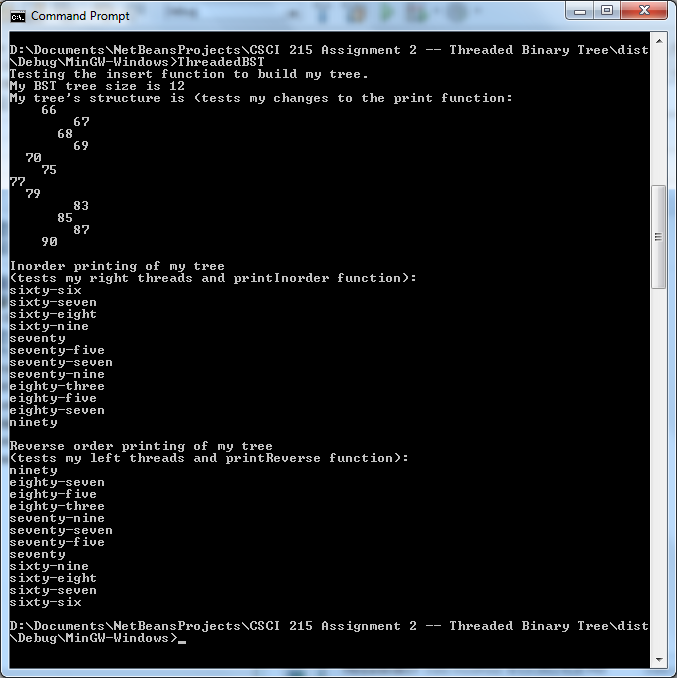
## Assignment Submission:

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

* Any changed or new source code files
* Your executable\*\*\*
* Word document describing your approach to solving this problem.
* Test documentation – you only get credit for the functions you have demonstrated work. Be sure that the documentation 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 you are only required to test the following combination: <int, string>.

\*\*\* If you completed your assignment using Visual Studios then give me your entire VS project. That way, if I have questions about its performance, I can easily rebuild your program myself.

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



Put all your source files, test files (files that showed your program in operation) executable, and Word document into a zip file using the following naming convention:

Student\_Name\_ThreadedBST.zip

Your assignment is due (see Blackboard and Course Schedule). **Late assignments will not be accepted.**

## Rubrics:

* **Program must run in order to get any points.** By run I mean you must at minimum:
  + Implement successor threads, and demonstrate
  + In-order printing without using recursion.
* Word document describing implementation included and sufficiently detailed in its description of the implementation. (10%)
* Successor threads implemented and demonstrated using printInorder without recursion. (60%)
* Predecessor threads implemented and demonstrated using printReverse without recursion (30%)
* You do not have to implement any of the remove or delete functions.

## 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 and be sure you understand how the author has implemented his BST. Then add your bit fields (or bool variables if you are not going to use bit fields) to the appropriate .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.

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