Assignment Objective: Build skills on C class creation and integration while implementing an **AVL Tree** for key-value pairs.

Requirements:

* Create a class called AVL that implements an AVL BST. Do this by copying BST.h to AVL.h, BST.cpp to AVL.cpp, and changing the class BST to AVL in the AVL.h and AVL.cpp files; don’t forget to include AVL.h, not BST.h, inside the AVL.cpp. The new members AVL will be:
  + Private variable:
    - bool doBal; // if true, then the bal() function shall perform its actions; otherwise, the bal() shall just set the height of the passed in node.
  + Private members:
    - void rotateLeft(node \* &p1) ; p1 is a node that is right heavy imbalanced.
    - void rotateRight(node \* &p1) ; p1 is a node that is left heavy imbalanced.
    - void bal(node \* &p) to balance p, if doBal is set, and if p is not NULL. If !doBal AND p != NULL, just set the height of p. If doBal and p != NULL, balance p as described in class.
    - int height(node \*p) const; // returns the height of p if it is real; otherwise 0 if it is NULL
    - int calcHeight(node \*p) to return the calculation of p’s height based on the height of its children; this is NOT a recursive call
  + Public members:
    - Modify AVL() to be AVL(bool doBal) and capture the value of that parameter in the doBal member variable.
    - Modify printIt(node \*p, int &index) to print the height of each node and its “index”; see the AVLcorrectOutput.txt file for the format.
    - Otherwise, there are no new members.
* Implement, if not already there, a non-member function “int max(int a, int b)” that returns the maximum value of ***a*** and ***b***.
* You must not use any other data structure, whether built-in or otherwise.
* Demonstrate that the AVL data structure works:
  + Create the appropriate Makefile
  + make AVL
  + Run your program as follows:
    - Run the program with balancing enabled:

./AVL 1 < AVLinput.txt > AVLoutput-1.txt

* + - Run the program with balancing disabled:

./AVL 0 < AVLinput.txt > AVLoutput-0.txt

// wait for it to finish; it will be a while

* + Compare AVLoutput-1.txt to AVLcorrectOutput.txt.
  + Observe the difference in the runtime between AVL 0 and AVL 1.
* Deliverables:
  + **Into D2L put a zip file containing:**
    - AVL.h, AVL.cpp, AVLoutput-0.txt, and AVLoutput-1.txt
    - DO NOT put a project into D2L
  + Turn in a hardcopy of your AVL.h, AVL.cpp, AVLoutput-0.txt and AVLoutput-2.txt files.