Purpose: Update a Binary Tree Class
Points: 25
Due: 9 AM on Wednesday, April 26, 2017
Overview: Update the Tree_ADT example from class to handle multiple data. The updated class will also permit entries to be changed.

Update the TreeNode class with a new data member `count`, of type int, that represents how many copies of the `info` data member are stored in the tree. This equivalence will be based upon the definition of the equality (==) operator for the instantiation’s template argument.

This requires updating the TreeNode class’s constructor. In the Tree_ADT class, you need to update some (or maybe all) functions. For example, the insertion function must be updated to check for equality between a TreeNode’s data and the new data and increment. **Every update must be noted with comment(s).**

One significant issue is how to access the counter in concert with the node data. Devise a solution and thoroughly document it in the TreeNode class declaration of the counter and also accessing the count in concert with the data in the Tree_ADT member functions.

Then, update the application as follows:
- Remove: If the value to be removed is in the tree and has multiples, ask the user whether they wish to remove one copy or all and act accordingly. If it is there once, just remove it.
- Print: Values that have multiples must have the counter printed as well. One possibility is to put the multiplicity in parenthesis ( ).

Next, add a function to the Tree_ADT class (and an accompanying option C)hange to the menu in the application) that takes two values, representing a value in the tree and the value to change it to. For example, if MyTree is a Tree_ADT object, then

MyTree.change(23,45)

will change the value 23 to 45. This requires addressing several issues:
- The precondition of this function is that the value to change WILL be found. Therefore, you must check for the value to be changed in the application before calling the change() member function.
- If (and only if) changing this value will violate the binary search tree invariant, you must remove the value from the tree and reinsert it.
  - This requires knowing the value of the parent and children of the node containing the value to be changed.
- If the value to be changed has multiples, then decrement the counter and insert the destination value.

Notes:
- For the change option, if you automatically delete and reinsert the value, your project will be penalized three letter grades.
- The C)hange subfunction in the application will prompt for the value to find and, if it is in the tree, the value to change it to. If not found, report that and go back to the menu. DO NOT READ BOTH VALUES AT ONCE.
- Your program will be tested with a script. The inputs must conform precisely to the specs. Letter inputs must be handled case insensitive. Two letter grade penalty if your program does not work with my scripts. TEST!!!
- Update the id and function comment blocks to meet both the standards in the code documentation document and the requirements for creating a Doxygen site.
- All files are provided in the project’s directory on acad, and also on the course website.

Turnin:
Same files you were provided. Include the link to your Doxygen site in a file, named exactly readme.txt, where you may provide bug reports, design decisions, etc;