CS 237 Programming Project 4 Kutztown

Purpose: Update a binary tree class to handle multiple datum
Points: 25
Deadline: Sunday, April 28, 2024 @ 10 AM. Submit late at your own risk.
Overview: Update the BinarySearchTree example from class to handle multiple data.

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 BinarySearchTree 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 (generically) and increment the counter if they match.

One significant issue is how to access the counter in concert with the node data. Thoroughly document issues that arise along with how they were addressed in your readme.

You will update the **application** as follows:

- Remove: If the value to be removed is in the tree and has multiples, give the user a choice to remove one copy or all (letters O or A) and act accordingly. If it has but one copy, remove it without inquiry.
- Print: Values that have multiples must have the counter printed as well. One possibility is to put the multiplicity in parenthesis ().

Notes:

- **Every update must be noted with comment(s).** Prompts are provided in the code.
- Your program will be tested with a script. The inputs must conform precisely to the specs. Letter inputs must be handled case insensitive. Your program won't be graded if it doesn't run properly formed scripts. Read the specs, follow them, and TEST!!!
    - One sample script will be provided
- For remove, when there are multiples of the item to be removed, the menu must accept either A or O, case insensitive. If there is only one copy of the element, there is **NO** interaction.
- A demo is provided. If you have a question about how the program runs, check the demo first.
- **Create drawings before coding.** Think about insertion/deletion cases. Support will be instructed to require drawings before discussing code. They will of course assist with creating drawings. So will the course instructor.
- Create a Doxygen site that has a mainpage with a description, a \file description in the file list for each file, a \class for each class, and a \brief for each function.
    - Documenting provided/updated code and mainpages are a point of emphasis in this project.
    - Create/update existing id and function comment blocks to create a robust Doxygen site.
        - Place existing material appropriate for the Doxygen site in Doxygen comments.
- All files are provided in the project's directory on acad and on the course website.

Deliverables:

D2L: File named exactly **readme** of type txt, doc, or pdf, containing

- Descriptions of issues with the multiplicity counter
    - Each issue that arose
    - Steps taken to address the issue
- bug reports, design decisions, etc;
- Doxygen link

Turnin: Same files you were provided, updated.

- Submit at least two input scripts that you created to test your project. Points off for failing to submit or submitting useless examples. Provide scripts that display thoughtful testing.