CPS305 Lab 3

Topics:

- BST, AVL

Files:

- Lab3.py
- Incorrectly named files will receive a mark of zero.

Submit Files:

- Submit file through D2L
- Your file should not have any print statements in it.
- You will lose a mark for every print statements.
- There is a getName() function in the Lab3.py file.
- Change the output string to your last and first name.
- Your name should be EXACTLY as written on D2L.
- Incorrectly filed out getName function result in a mark of zero.

Lab Description:

In this lab, you are implementing a linked solution of your own BST and AVL. You are given Python class templates of two classes, MyBST and MyAVL Some of the functionalities are given and you are to complete the rest as indicated by the requirements.

Requirements:

- First, create a method called getName() that returns your name. If this does not work, you will receive a mark of zero.
- A class called MyBST is given. Write the method remove() to meet the definitions of a Binary Search Tree. To properly implement the remove() method, you will also need to implement the methods updateHeight() and copySmallest() or copyLargest(). The method updateHeight() updates the height of the node. Depending on which implementation you choose, you can implement copySmallest() or copyLargest() which copies the smallest or largest nodes from the left subtree or right subtree of the removed node. Write the method called __contains__(x) to return true if x is in the tree.
- The following methods: getLeft(), getRight(), getData(), and insert() are given. Do not change them. Although the insert() method is given, it utilizes the updateHeight() method so that must be implemented correctly.
- The method __init__() is also given. You may add to this. Note the class variable "promote_right." Set it to true if you plan on implementing the convention that promotes from the right subtree when removing or set it to false otherwise.
- A class called MyAVL that extends MyBST is given. You must implement a new remove() method to override the MyBST remove() method. You also must implement the methods reBalance() and getBalanceFactor().
- The getBalanceFactor() method returns the balance factor of the current node.
- The reBalance() method should rebalance the AVL subtree that is out of balance and return the newly balanced subtree. This method will use the functions rightRotate() and leftRotate(), which are given.

Restrictions:

- This lab can be solved using brute force. Of course, we do not want that and thus, when you run the tester, it will timeout after a while and if your lab does not finish you would not get the marks.
- You cannot use Python's list in your implementation. This includes the methods appends, pop, etc. This lab requires linked implementations of BST and AVL

Testing Your Code:

You can test your code with the tester. The tester is not a be all and end all. You should rely on your own testing before using the tester. Note that the lab should be done on Python 3 and the tester only works on Python 3. The tester is named Lab3Tester.py. Your file must be named correctly and be in the same folder of the tester. Here is an example of executing it on the command line (note that the $ is representative of the beginning of the new line).

```
$ python3 Lab3Teser.py
```