Assignment 3, Part 2 of 2
AVL Tree Implementation

Instructor: Homeyra Pourmohammadali
MTE140 - Data Structures and Algorithms
Winter 2021 (Online)
University of Waterloo
Due: 11:00 PM, Wed, April 7th, 2021.

Purpose of this assignment

In this assignment, you will practice your knowledge about **tree** by implementing a data type called **AVL tree**. The header file avl-tree.h, which is explained below, provides the structure of the AVLTree class with declarations of member functions.

Note that the AVLTree class is an extension through inheritance of your BinarySearchTree class. If you are not familiar with the concept of inheritance, e.g., how to define and use inherited variables and functions, read a simple tutorial here: https://msdn.microsoft.com/en-us/library/84eaw35x.aspx.

Instruction

Sign in to GitLab and verify that you have a project set up for your Assignment 3 (A3) at https://git.uwaterloo.ca/mte140-1211/a3/WATIAM_ID with the following files.

```
YOUR-WATIAM-ID
 |__ CMakeLists.txt
 |__ README.md
 |__ user.yml
 |__ binary-search-tree.cpp
 |__ binary-search-tree.h
 |__ avl-tree.cpp
 |__ avl-tree.h
 |__ test.cpp
```

For this part of assignment, you need to modify avl-tree.cpp and maybe also avl-tree.h if necessary. You can design your own test case and code in test.cpp. It is optional and we will not grade this file.

You can use the same procedures in Assignment 0 to pull, edit, build, commit, and push your repo.

Description

You need to implement the new AVL versions of insert and remove operations that keep the tree balanced. Place your code into avl-tree.cpp. When defining the AVL version of them, you may want to call the Binary Search Tree version of insert and remove inherited from your Binary Search Tree class, using BinarySearchTree::insert or BinarySearchTree::remove, to insert or remove the node from the tree, respectively. After that, you need to ensure that the tree is kept balanced by applying the appropriate AVL tree rotations.

Do not modify the signatures of the insert and remove functions to ensure that the test cases pass. The AVL tree should be kept balanced after calling just insert or remove, so any re-balancing operations should happen within these functions rather than outside of them. You may create additional functions and/or attributes in the .h and .cpp files to help complete the tasks, if needed.

Marking

We will try different inputs and check your output. We will only test your program with syntactically and semantically correct inputs.

Part 2 counts **40%** of Assignment 3, which is 40 points in total.

Your program runs and does not crash during the test: + 10
Passes Test Cases: + 5 each, in total of 30