EECS 560 Lab 06: Self-Balancing AVL Binary Search Tree (AVL-BST)

Objective

- Get familiar with binary search tree implementation with C++.
- Get familiar with AVL tree implementation with C++.

Specifications

1. Implement the self-balancing AVL binary search tree data structure as the `MyBST` class.
2. Implement a function `bool lowestCommonAncestor(const ComparableType& x, const ComparableType& y, ComparableType& lca)` that calculates the Lowest Common Ancestor (LCA) of the two input data elements `x` and `y`. If both data elements are in the AVL-BST, return true and store their LCA in `lca`. Otherwise, return false.

Testing and Grading

We will test your implementation using the tester main function posted online. The posted input and output examples should be used for a testing purpose, while we will use another set of inputs for grading. Your code will be compiled under Ubuntu 20.04 LTS using g++ version 9.3.0 (default) with C++11 standard.

Your final score will be the determined by the success percentage of your program when fed with many random inputs. Note that if your code does not compile (together with our tester main function), you will receive 0. Therefore, it is very important that you ensure your implementation can be successfully compiled before submission.

Submission and Deadline

Please submit your implementation as a single .h file, with a file name "MyBST_[YourKUID].h". For example, if my KU ID is c123z456, my submission will be a single file named "MyBST_c124z456.h". Submissions that do not comply with the naming specification will not be graded. All submission will go through Blackboard. The deadline is October 28th, 2022 11:59PM.