CptS 223 – Advanced Data Structures in C++
Programming Assignment 2: Adelson-Velskii and Landis (AVL) Trees

I. Learner Objectives:

At the conclusion of this programming assignment, participants should be able to:

Develop a C++ program that applies an AVL tree data structure
Design and implement an AVL tree data structure

II. Prerequisites:

Before starting this programming assignment, participants should be able to:

Apply and implement class templates
Design, implement, and test medium programs in C++
Edit, build, debug, and run programs through a Linux environment

III. Overview & Requirements:

For this assignment, you are going to design and implement an AVL tree data structure.

What is required?

(20 pts) You are required to have class *template* `AVLNode`. The corresponding implementation for `AVLNode` must be placed into a single header file called `AVLNode.h`. The class must contain **at least** the following *public* member functions:

- (5 pts) necessary constructors for your implementation, destructor

This class must also contain **at least** the following data members:

- (5 pts) the data (a generic T type, which allows any numerical data)
- (5 pts) a left pointer to an `AVLNode`, a right pointer to an `AVLNode`
- (5 pts) a height (int type) that records the current height of the node

---

(60 pts) You are required to have class *template* `AVLTree`. The corresponding implementation for `AVLTree` must be placed into a single header file called `AVLTree.h`. The class must contain **at least** the following *public* member functions:

- (5 pts) necessary constructors for your implementation, destructor
- (10 pts) height () – returns the height of the tree

- (20 pts) insert () – inserts a value (type T) into the tree – must abide by AVL balancing properties
- (10 pts) contains () – given a value (type T), returns 1 if it is in the tree; 0 otherwise.
- (10 pts) validate () – returns 1 if this tree is a balanced tree; 0 otherwise. Recall this: height of left and right subtrees at every node in a AVL tree differ by at most 1.

This class must also contain **at least** the following data member:

- (5 pts) a pointer to the root `AVLNode`

(10 points) Verification of your project by building it using cmake
(5 pts) Write CMakeLists.txt correctly.
(5 pts) Use cmake to build project successfully.
Cmake is a cross-platform make tool. Building your project by cmake can verify the cross-platform feature of your project.
How to cmake: check the document on Canvas that contains the installation instruction and examples of cmake.

(10 pts + 10 bonus pts*) Test cases. In your `main.cpp`, you are required to create 3 data sequences (use whatever data structure you like to store the data). These sequences all contain the odd numbers between 1 – 100. So every sequence has 50 numbers. Numbers in 3 sequences are put in three different orders (1) ascending order (2) descending order (3) random order (you could use std::shuffle)

Now in the main function,
1. (5 pts) Create three AVL trees and insert the three data sequences into three AVL trees, separately. So each AVL tree will have 50 nodes.
2. (5 pts) Print the height of each AVL tree. (hint: what is the expected output?)
3. (5 bonus pts) Print the result of validate () for each AVL tree.
4. (5 bonus pts) For each AVL tree, loop over Number 1 – 100 and call contains() function for each number. For odd number, contains() should return 1; for even number, it should return 0. If the result is not as expected, your code should print "My AVL tree implementation is wrong".

* Bonus pts: the total points (including bonus) of PA2 cannot exceed 100 pts. For example, you earn 100 standard pts and 10 bonus pts, your final grade of

PA2 will be 100 pts. However, if you lose 5 pts from the previous 100 pts and earn 10 bonus pts at the same time, your final grade will be 100 pts.

IV. Submitting Assignments (either Canvas or Github):

1. Option 1: Github

1) On your local file system, and inside of your git repo for the class, create a new branch called PA2. Available to the branch create a new directory called PA2. Place all PA2 files in the directory. All files for PA2 must be added, committed, and pushed to the PA2 branch of your private GitHub repo created in PA1.
2) Submission: You MUST submit a URL link to the branch of your private GitHub repository. Please make sure the instructor and two TAs (GitHub account listed in Syllabus) are the collaborators of your repository. Otherwise, we won't be able to see your repository. DO NOT CREATE NEW REPO.
3) Do not push new commits the branch after you submit your URL to Canvas otherwise it might be considered as late submission.

2. Option 2: Canvas

Zip all source files (including CMakeLists.txt, etc., if any) and upload it to Canvas.

V. Grading Guidelines:

This assignment is worth 100 points + 10 bonus points.