# CSUC CSCI-311 - Algorithms and Data Structures
# Lab Assignment 5

Goal: 1) practice using AVL trees in C++.

In this lab we will build our own AVLTree class by implementing the other methods in the AVLTree_skeleton.cpp file.

Submission: C++ solutions to these problems should be written in a file called AVLTree.cpp (a skeleton is available on Canvas along with AVLTree.h, AVLNode.h, AVLNode.cpp, and AVLTreeDriver.cpp) and submitted on inginious (inginious.csuchico.edu)

Coding Style: Note that your submission should use good C++ coding style and should include appropriate comments (for readability and maintainability). Specifically, your code must follow common C++ coding conventions for Naming, Indentation, and Comments. Points will be deducted if these are not present.

Collaboration: There will be time in lab to discuss these problems in small groups and I highly encourage you to collaborate with one another outside of class. However, you must write up your own solutions **independently** of one another. In addition, do not post solutions in any way. Also, please include a list of the people you work with in a comment section at the top of your submission.

In addition, Notes 07.02 and 07.03 will be very helpful for completing Lab 5.

Have fun!

| | |
|---|---|
| Assignment Date: | **Oct 18, 2024** |
| Due Date: | **11:59pm on Oct 25, 2024** |
| Grace Due Date: | **11:59pm on Oct 28, 2024** |

Grading Notes: 1) Assignment is due on the Due Date. 2) For each day late after the Due Date, there will be 10% penalty on the assignment's grades. 3) Submission is not accepted after the Grace Due Date. In other words, you will receive zero pts if your submission is not received by the Grace Due Date.

NOTE: This lab is hard! Start early, and please read lecture notes 07.02 and 07.03 before doing this lab.

Grading: Coding Style 5 pts, Test Cases 95 pts, Total 100 pts.

Assignment Questions (95 pts)

1. Implement a class constructor along with getters for the root and the size of the tree.
2. Implement the rotate left method for AVL trees.
3. Implement the rotate right method for AVL trees.
4. Implement the rotate left-right method for AVL trees.
5. Implement the rotate right-left method for AVL trees.
6. Implement the rebalance method for AVL trees.
7. Implement pre-order, in-order, and post-order traversals for AVL trees.

8. Implement the insert method for AVL trees.
9. Implement the search method for AVL trees.
10. Implement the minimum method for AVL trees.
11. Implement the maximum method for AVL trees.
12. Implement the delete method for AVL trees.