**1310 PROGRAM 4 / Fall 2020**

**AVL Trees**



# important dates

**Assign Date:** Friday, October 29, 2020

**Due Date:** Monday, November 23, 2020 by 11:59pm

**(can be turned in up to 2 days late at a penalty of 10 points off per day late)**

# Description

Create a program that will read Computer Science professor data from a text file, create Professor objects from this data (using the Professor class provided for you), insert the data into an AVL Tree (via the AVL Tree class that you create), and then allow the user to search for a professor by name to view their information.

The main advantage of an AVL tree is searching because searching can be done incredibly fast.

Follow the specifications below to get full credit on this assignment.

# Files to Be Included In Your Submission

* **AVLTree.h** – specification file for your AVLTree class.
* **AVLTree.cpp** – implementation file for your AVLTree class
* **Professor.h** – provided for you
* **ProfessorData.txt** – provided for you
* **Driver.cpp** – contains the main function and the readFromFile function

# main program (DRIVER.cpp)

### Main Function

1. Create the AVLTree object.
2. Call the readFromFile function, sending the address of the AVLTree to the function
3. In a loop, do the following:
   1. Ask the user which CS professor they want details about
   2. Print out the professor’s name in alphabetical order by calling the AVLTree object’s displayInOrder function.
   3. Get the string name from the user of the professor.
   4. Call the AVLTree object’s searchNode function, sending the name the user entered to this function. This function should return a pointer to the Professor object with that name, or NULL if the professor doesn’t exist in the AVL Tree.
      1. Print “That teacher is not in the AVL Tree.” If the function returned NULL.
      2. Print the Professor object if the function did not return NULL. Note that the Professor class has an overloaded << operator.
   5. Ask the user if they want to search for another professor (let user enter y or n). If user enters ‘n’, then end the program

### ReadFromFile Function

1. Open the ProfessorData.txt file.
2. If it can’t be opened, print “Sorry! Can’t read professor data from file!” and end the program.
3. Otherwise, use a loop to:
   1. Read each piece of data from the file, which is separated by dollar signs. Use getline with the ‘$’ delimeter. The order the data is in is the professor’s name, the course they teach, then there is a series of zeroes and ones that represent yes or no. The zero means no and the 1 means yes.
   2. Create a new Professor object
   3. Call the AVLTree object’s insertNode function, sending the name of the professor and the Professor object to this function.
   4. Keep track of how many professors you add to the AVL Tree during this loop
4. Print how many professors were added from the file after the loop ends.

# AVLTree.h & AVLTREE.cpp

### Attributes:

* A TreeNode structure containing the following members:
  + string name (professor’s name)
  + Professor \*prof; (Memory address of a Professor object)
  + TreeNode \*left;
  + TreeNode \*right;
* A pointer to a TreeNode called root

### Private Member Functions:

* void insert(TreeNode \*&, TreeNode \*&)  
  Recursive function called by the public insertNode function. This function also calls the balance function.
* void destroySubTree(TreeNode \*)  
  This function is called by the destructor. It deletes all nodes in the tree.
* void displayInOrder(TreeNode \*) const  
  This function displays the Professor’s names in the subtree pointed to by the TreeNode pointer sent to this function, via inorder traversal.
* int height(TreeNode \*nodePtr)  
  Find the height of the AVL Tree and return it.
* int diff(TreeNode \*nodePtr)  
  Find the HEIGHT DIFFERENCE between the left & right subtrees of the given node and return. (this is the balance factor)
* void balance(TreeNode \*&nodePtr)  
  Balance the tree into an AVL Tree
* TreeNode\* l\_rotation(TreeNode \*parent)  
  Performs a Left rotation on the parent node
* TreeNode\* r\_rotation(TreeNode \*parent)  
  Performs a Right rotation on the parent node
* TreeNode\* lr\_rotation(TreeNode \*parent)  
  Performs a Left-Right rotation on the parent node
* TreeNode\* rl\_rotation(TreeNode \*parent)  
  Performs a Right-Left rotation on the parent node

### Public Member Functions:

* AVLTree()  
  AVLTree class constructor, which sets the root pointer to NULL
* ~AVLTree()  
  AVLTree class destructor, which calls the private recursive destroySubTree function (sending the root)
* void insertNode(string, Professor\*)  
  This function is called by the Driver and receives the Professor’s name as the 1st parameter and a pointer to the Professor object ans the 2nd parameter. This function calls the private recursive insert function.
* Professor\* searchNode(string)  
  This function determines if the Professor’s name sent to this function is present in the tree. If so, the function returns the pointer to the Professor object. Otherwise, it returns NULL.
* void displayInOrder() const  
  This function calls the private displayInOrder function, sending the root.

# Sample Output (I used the creature class for my class)

There were 11 professors added from the file.

Which Computer Science professor do you want details about?

Andy Bernard

Angela Martin

Dwight Schrute

Jim Halpert

Kelly Kapoor

Kevin Malone

Meredith Palmer

Michael Scott

Oscar Nunez

Pam Beesly

Phyllis Vance

TYPE PROFESSOR NAME: Jim Halpert

Professor: Jim Halpert

Course: CSC1310

Clear Grading Criteria: yes

Provides Good Feedback: yes

Caring: yes

Reachable Outside of Class: no

Tough Grader: no

Lecture Heavy: no

Attendance Mandatory: no

Would you like to search another?(y/n) y

Which Computer Science professor do you want details about?

Andy Bernard

Angela Martin

Dwight Schrute

Jim Halpert

Kelly Kapoor

Kevin Malone

Meredith Palmer

Michael Scott

Oscar Nunez

Pam Beesly

Phyllis Vance

TYPE PROFESSOR NAME: Michael Scott

Professor: Michael Scott

Course: CSC1310

Clear Grading Criteria: no

Provides Good Feedback: no

Caring: yes

Reachable Outside of Class: yes

Tough Grader: no

Lecture Heavy: yes

Attendance Mandatory: yes

Would you like to search another?(y/n) y

Which Computer Science professor do you want details about?

Andy Bernard

Angela Martin

Dwight Schrute

Jim Halpert

Kelly Kapoor

Kevin Malone

Meredith Palmer

Michael Scott

Oscar Nunez

Pam Beesly

Phyllis Vance

TYPE PROFESSOR NAME: Angela Martin

Professor: Angela Martin

Course: CSC1300

Clear Grading Criteria: yes

Provides Good Feedback: no

Caring: no

Reachable Outside of Class: no

Tough Grader: yes

Lecture Heavy: yes

Attendance Mandatory: yes

Would you like to search another?(y/n) y

Which Computer Science professor do you want details about?

Andy Bernard

Angela Martin

Dwight Schrute

Jim Halpert

Kelly Kapoor

Kevin Malone

Meredith Palmer

Michael Scott

Oscar Nunez

Pam Beesly

Phyllis Vance

TYPE PROFESSOR NAME: April Crockett

That teacher is not in the AVL Tree.

Would you like to search another?(y/n) n