**CSL201: Assignment 4**

**Due Date :October 2, 2013**

**Storing employee records in an hierarchical structure supported by an AVL tree**

In this assignment, we shall extend the previous assignment by making the operations more efficient. For the purpose of this assignment, you can assume that no two employees have the same name.

The details of this assignment are more or less the same as in Assignment 3, and you have to carry out the same set of operations as in Assignment 3. However, now you will make an AVL tree on employee names. This AVL tree will also store for each employee, where its corresponding node in the original employee tree is. Whenever you need to delete, search or insert a node in the employee tree, you will first use this AVL tree to quickly find out the location of the employee in the employee tree. Changes from the previous assignment have been marked in the bold below.

Your program should be able to read data from a file. The file contains some integers and some words. The first entry is guaranteed to be an integer. Depending on the value of the integer your program should do the following:

1 Read the name of the CEO. Keep this person at level 1.

2 Read the names in pairs. For each pair (A,B) add the employee A in the tree so that he reports to immediate boss B. Note that automatically level of A will be one more than that of B.

**3 Read the integer L. Print the names of all employees at Level L, and the employees reporting directly to them. Thus if there are two employees Y and Z at level L, with R,S,T reporting directly to Y and M,N reporting directly to Z then print the names in the following format:**

**Y: R S T**

**Z: M N**

4 Read the name K. Print K and within curly braces print the names of all the employees who happen to be boss of K, starting with the immediate boss.

5 Read the name pair (C D). Remove C from the tree and rearrange the employee records such that all persons working under C will now work under D. If no one is reporting to C then simply remove C.

6 Read the name pair (G F). Print names G and F within curly braces. Print the name of the employee who is common boss of G and F at the lowest hierarchy in the company. There will be many bosses of G and F but this boss will be at the highest level amongst all the bosses.

7 print a line of character “-“.

8 Print your name and entry number and exit the program.

**9 Print “AVL”. Print “LEFT” followed by number of nodes in the left subtree of your AVL tree. Print “RIGHT” followed by number of nodes in the right subtree. In the next line print the pre-order traversal of the AVL tree.**

Credit will be given to choice of proper data-structures and efficiency. For example, in operation 5 above, one should not just search the entire tree to look for the name of the employee C. Your program should also catch errors, for example if in operation 2 above, there is no employee with name B, then it should say it is an error. Use the notions of exceptions in C++ to implement error checking.