Graded Assignment 2 – Data Structures

For this assignment, you will solve problems based on the concepts of Trees.

Instructions

- **Your C program should carry comment statements at the top indicating your name and a statement confirming that the program is completely your own work and that no part of it has been copied from anybody else or downloaded from the net.**
- **Assignment submitted after due date will not be evaluated and a score of zero will be awarded for this assignment.**
- **You have to submit .c file via BB.**

Due Date: Midnight, April 25, 2025.

Submitting this Assignment

- You will submit (upload) this assignment in Blackboard. Email submissions will not be accepted at any cost.
- Name the file as "GA1_2025_John_Doe.c" in case your name is "John Doe".

Grading Criteria

This assignment has absolute 7 points. Viva will be conducted for suspected cases of plagiarism.

Question:

The first task in this assignment is to create a binary tree for an organization. Each node of the tree represents an employee of the organization, and the tree represents hierarchy of employees reporting to their immediate bosses. Each employee would be represented by an uppercase alphabet in the datafile *employee.txt*. A tuple (node left-child right-child) would represent a two member team of left-child and right-child reporting to boss at node value. Thus the tuple (P D F) would mean that there is a team of left child D and right child F, both reporting to boss P. Any missing member would be represented by X. Thus the tuple (M B X) would mean that in this team employee B is the sole member reporting to boss M. Further the tuple (K X X) would mean that K is not a boss as no one reports to him or her.

On the datafile, the tuples are going to be specified in an arbitrary order, i.e. you will not specifically be told who the CEO of the organization is. You will have to figure out the root of the tree by examining the various tuples.

Other tasks:

task 2: Print the inorder traversal of the tree

task 3: Write code to print the height of the tree.

task 4: Add code to scan the tree to print the number of bosses in the organization.

task 5:

Add code to check if the tree satisfies the condition of being a binary search tree. To check the BST property, use the alphabetical order of the nodes. Thus, for example, T > M > J > D >A. If it is a BST, print a statement that the tree is a BST, indicating the CEO of the organization by using *the alphabet value* at the root.

If the tree is not a binary search tree, your program should print that it failed the BST property, and point out where the condition failed, by printing *the alphabet value* at the failing node.

task 6:

The Board Members of the organization decide do some role shuffling of the employees. Some employees get switched from boss positions to team member positions and also it may happen the other way around for some employees. New bosses are created and some of the employees now report to new bosses. These are implemented here through the function indicated below. In this task 6, you will call the following function on the root of the tree. Here it assumes the Tnode structure used in last lab class. If you are using a different node structure, feel free to modify it for the node structure being used by you. Print the preorder traversal of the new tree.

```
struct node *specialFunction(struct Tnode *root){
        struct node *alpha;
        struct node *omicron;
        alpha = root;
        omicron = alpha->right;
        alpha->right = omicron->left;
        omicron->left = alpha;
        return omicron;
    }
```

task 7:

In this task, you will be specified an alphabet *nnn*. Go through the tree to check if *nnn* is actually an employee of the organization. Report accordingly. If *nnn* is found on the tree, then print the name of *nnn*'s team mate.

If *nnn* happens to be the sole team member, that is, there is no team mate, print a message indicating the same.

Repeat the above for another alphabet *ppp*.

Format of input datafile:

The first 2 characters on the file would represent *nnn* and *ppp*. These will be followed by node tuples data (as explained above). It is guaranteed that there will be no small case letters on the datafile. All data values are going to be in capital letters.

Sample input datafile:

```
P T  N L P  J F N  P X T  F X X  L X X  T X X
```

Here the first two letters P and T correspond to *nnn* and *ppp*. N L P is the first tuple.

Sample output:

```
Inorder Traversal:

F J L N P T

Tree Height:    4

Number of Bosses: 3

It is a BST tree with CEO: J

Preorder after shuffling:

N J F L P T

Confirmed that P is an employee.

Team mate of P: J

Confirmed that T is an employee.

T is sole team member.
```