ANKARA UNIVERSITY

COM102B-120

Fall 2018-19 (Spring)

Programming Assignment 2

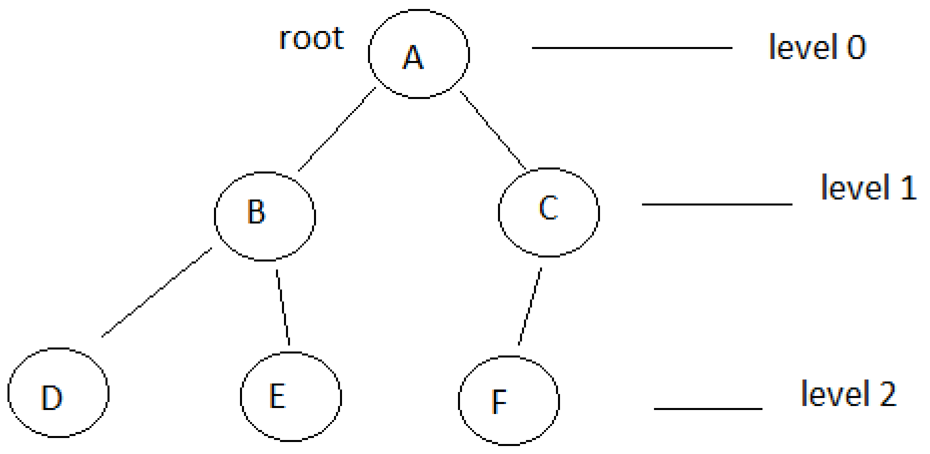
Submission Deadline: 10/04/2019, 23:55

In this assignment, you will implement a class that contains a tree structure, which is depicted below. You will design your class with necessary member functions such that, your class object will keep a sequence of words from the std. input. Node structure and tree class data members are depicted below:

```
struct Node{
        string word;
        Node* left;
        Node* right;
        Node* father;
};

class tree
{
        Node* root;
…
}
```

All structure objects that is linked to the tree should be allocated dynamically using the **new operator**. Assume that you read 6 words, in this order: A B C D E F. The structure of the tree will be constructed such that the following node relations will be formed between the nodes:



Each incoming node will be placed in the next empty slot in a tree level, from left to right order. When that level is filled, the incoming node is stored to the leftmost position of the next level. In this tree, the node at the top of the tree is called the **root** node and this node is represented as a member of your class object (of tree class). Each node can have at most two child nodes. The one on the left is called as left child, on the right is called as the right child. The tree definition is recursive, i.e. the left child of A, is also a tree that is rooted at B with the left child D and the right child E. You need to write necessary functions that inserts a node into the tree structure. You will also need to write a set of member functions that implements the traversal in the tree. You will implement the following four member functions. Note that each function has an ID, which will be provided in the input format to specify the requested function.

Function id [1]: traverses and prints the words in the tree in *pre-order form*:
(1) print the word at the root, (2) print the word at the left subtree in pre-order, (3) print the word at the right subtree in pre-order.
Ex: The output for the example tree will be: A B D E C F

Function id [2]: traverses and prints the words in the tree in *post-order form*:
(1) print the word at the left subtree in post-order, (2) print the word at the right subtree in post-order, (3) print the word at the root.
Ex: The output for the example tree will be: D E B F C A

Function id [3]: traverses and prints the words in the tree in *in-order form*:
(1) print the word at the left subtree in in-order, (2) print the word at the root, (3) print the word at the right subtree in in-order.
Ex: The output for the example tree will be: D B E A F C

Function id [4]: prints the nodes that does not have any child, from left to right.
Ex: The output for the example tree will be: D E F

The function id will be provided as input to your program; you will call the corresponding function. Functions can be queried in any order. Your program will be terminated when **-1** is given from the input as a function id.

The incoming word list will be terminated with the **end** string.

Input format:
[<words> **WS**]
end
[<function_id>**WS**]
-1

[..] encodes at least one relation.
WS: represents white space characters

Ouput format:

*******Newline**
[<resultant function output>**Newline]**
Done!

Please check the sample I/O files, before starting your implementations.

Testing:

We provide sample input/output text files for you to test your codes at Ubuntu. We recommend you to use *input redirection* mechanism of your operating system to test your programs. For example, if your executable is called as **PA2**, redirect the **input.txt** file to standard input using **<** operator and redirect your outputs to a file using **>** operator such as:

```
> ./PA2<input.txt>output.txt
```

This kind of execution enables your programs to read inputs from a file without writing any file related functions. In other words, **cin** reads data from the redirected files instead of the std. input in this way (e.g. keyboard).

Warning: Any form of code copying, including the copies from the internet, is strictly prohibited. If we determine similarities between your codes with any other students in the class, it will be treated as cheating and will be punished. So, you would increase the risk of cheating when you see somebody else's code directly or see a solution in the internet (i.e. somebody else might have also copied the same code from the internet like you, so both of these codes will be evaluated as copies, since they both copy from an external source. Such attempts will always be considered as cheating). You are supposed to write the solution by yourselves.

Please test your programs with the given I/O files before submission.
Ask any questions related with the homework specs. to the course news forum in Moodle.
Follow the announcements about the submission of this homework from Moodle.

have fun ☺