ESS201: Programming II - Lab
C++ Programming Assignment: Composition

Jaya Sreevalsan Nair / IIIT Bangalore

Lab Date: 2019-10-23/Wednesday/13:30:00/IST
Submission Deadline: 2019-11-01/Friday/23:59:59/IST

This lab assignment is to get familiar with class composition.

Write a code to use classes to create a tree given its pre-order and in-order traversals. The executable, **mytree** output should be a modified post-order traversal and a customized output, in the post-order traversal. Conventionally, the traversals are:
Pre-order: (Parent, left-subtree, right-subtree)
Post-order: (Left-subtree, right-subtree, parent)
In-order: (Left-subtree, parent, right-subtree)

There must be a class for Node, with data members:
Node *_right, *_left, and Data *_data.
There must be a class for Data, with data members:
std::string _name, and int _index.
_index is the order of the node in pre-order traversal, where the indexing starts with a '0'.
The input includes:

1. The number of nodes in the tree,
2. the pre-order traversal using _name values of the nodes, and
3. the in-order traversal using _index values.

The pre-order traversal enables the construction of the nodes of the tree, as the data is available, along with the implicit _index values available from the ordering itself. Both the pre-order and in-order traversals are required for identifying the placement of the nodes in the tree.
The leaf nodes will have _right=_left=0 (null-pointer). In the absence of either the left or right subtree for a node, use the null-pointer to indicate the termination of the branch.

The std::ostream output of Data Class has to be overloaded such that _name and _index are outputted with tabs spacing in between two nodes:
Data d ;
std::cout << d ;
should be outputting:
std::cout << d.get_index() << "m" << d.get_name() << "\t" ;
-- using an overloaded operator << function, as a friend function.

The modification in the output is where each node is printed in std output using the overloaded function.

Assume you are working with a binary tree.

It is mandatory to write multiple files -- separate header and source files for each class; and a main.cpp to manage the input and output of the assignment.

Sample input and output for the tree, as shown in Figure 1.

```
$ ./mytree 9 Aa Ab Ac Ad Ae Af Ag Ah Ai 2 1 3 0 6 5 7 4 8
$ 2 Ac    3 Ad     1 Ab    6 Ag    7 Ah    5 Af    8 Ai    4 Ae    0 Aa
```

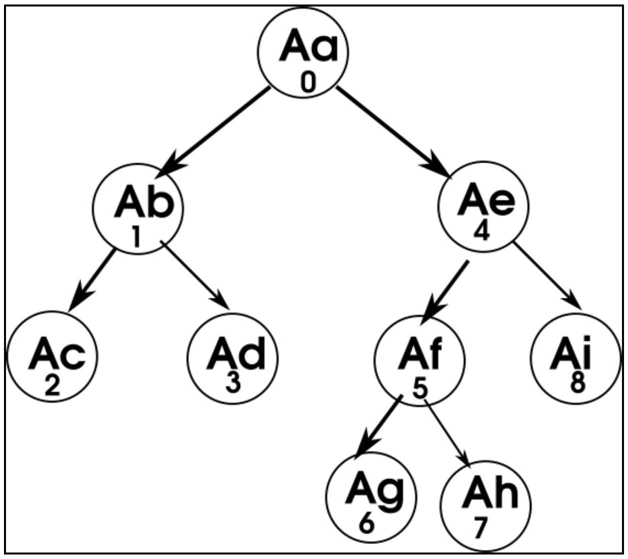

Figure 1: Tree constructed from sample input.