جـــامعة نيـويـورك أبـــوظبي
NEW YORK UNIVERSITY ABU DHABI

8

DATA STRUCTURES

Lab Exercise # 8: General Tree Implementation and Traversals

In this lab exercise, you will implement a General Tree data structure. The starter code includes the **Node** and **Tree** classes with some methods already implemented. Your task is to implement the following missing methods and to interact with the tree through a command-line interface. The goal is to understand tree operations, traversal techniques, and dynamic memory management in C++.

1. `void insert(Node* node, string child_name)` **(0.1 Points)**
   This method creates and inserts a new node with the given name as a child of the specified node. An error message must be printed if a child with the same name already exists in the current node.

2. `void remove(Node* node, string child_name)` **(0.1 Points)**
   This method removes a specific child node (along with its all children/subtree) from the given parent node. If the specified child node does not exist, it should print “child not found.”

3. `void isExternal(Node* node)` **(0.05 Points)**
   This method checks and returns True if a given node is an external (leaf) node, and False otherwise.

4. `void isInternal(Node* node)` **(0.05 Points)**
   This method checks and returns True if a given node is an internal (non-leaf) node, and False otherwise.

5. `int size(Node* node)` **(0.1 Points)**
   This method calculates and returns the number of nodes in the tree/sub-tree rooted at the given node.

6. `int depth(Node* node)` **(0.05 Points)**
   This method calculates the depth of the specified node.

7. `int height(Node* node)` **(0.1 Points)**
   This method calculates the height of the specified node.

8. `int treeHeight()` **(0.05 Points)**
   This method calculates the height (maximum depth) of the entire tree.

9. `void preorder(Node* node)` **(0.1 Points)**
   This method traverses and prints the tree in pre-order.

10. `void postorder(Node* node)` **(0.1 Points)**
    This method traverses and prints the tree in post-order.

11. `~Node()` **(0.1 Points)**
    The destructor for the Node class must delete the node and all its children. Ensure proper memory management to avoid memory leaks.

Comments: **(0.1 Points)**
Comments are a very important part of any program. You should always write comments in your code, even if not explicitly asked.

Desired Output:

```
> g++ lab8.cpp && ./a.out
>print
root
>insert d1
>insert d2
>print
root
├──d1
└──d2
>cd d1
>print
root
├──d1
└──d2
>insert d11
>print
root
├──d1
│   └──d11
└──d2
>preorder
root d1 d11 d2
>postorder
d11 d1 d2 root
>height
The height of the current node(d1) is: 1
>cd ..
>height
The height of the current node(root) is: 2
>treeHeight
The height(max-depth) of the Tree is: 2
>isExternal
Current node(root) is not an external node of the tree.
>cd d1
>remove d11
d11 has been deleted successfully.
>print
root
├──d1
└──d2
>height
The height of the current node(d1) is: 0
>treeHeight
The height(max-depth) of the Tree is: 1
>exit
d1 has been deleted successfully.
d2 has been deleted successfully.
root has been deleted successfully.
```

CODE OF CONDUCT

All assignments are graded, meaning we expect you to adhere to the academic integrity standards of NYU Abu Dhabi. To avoid any confusion regarding this, we will briefly state what is and isn't allowed when working on an assignment/lab-task.

Any documents and program code that you submit must be fully written by yourself. You can, of course, discuss your ideas with fellow students, as long as these discussions are restricted to general solution techniques. Put differently, these discussions should not be about concrete code you are writing, nor about specific results you wish to submit. When discussing an assignment with others, this should never lead to you possessing the complete or partial solution of others, regardless of whether the solution is in paper or digital form, and independent of who made the solution, meaning you are also not allowed to possess solutions by someone from a different year or course, by someone from another university, or code from the Internet, etc. This also implies that there is never a valid reason to share your code with fellow students, and that there is no valid reason to publish your code online in any form.

Every student is responsible for the work they submit. If there is any doubt during the grading about whether a student created the assignment themselves (e.g. if the solution matches that of others), we reserve the option to let the student explain why this is the case. In case doubts remain, or we decide to directly escalate the issue, the suspected violations will be reported to the academic administration according to the policies of NYU Abu Dhabi.

https://students.nyuad.nyu.edu/campus-life/community-standards/policies/academic-integrity/