SNU CSE: Data Structure

Due Date : 11 pm, May 15 2019

Assignment 3

Lecturer: Prof. Satti

1. Complete `MyTree.java`

   a. Construct `MyTree(String preorder, String inorder)`
      - Given the preorder and inorder traversal sequences of a binary tree, construct the binary tree, if possible and print `'Done'`; else, print `'Unaccepted input'`.
      - The traversal sequences are given as strings over an upper-case letter ('A' - 'Z') - each letter appears at most once.

   b. `String postorder()`
      - returns the post order traversal sequence of the tree (as a string)

   c. `String levelorder()`
      - returns the level order sequence of the tree (as a string)

<u>Note</u>

- Implement the tree using the linked representation.
- You may create more methods, data members, and classes as needed.
- You may use any Java built-in data structures
- The input file will not have empty spaces nor lines.
- Unmentioned exceptions/errors do not need to be handled.

<u>Example (running MyTree.java with myTreeInput.txt)</u>

```
Input : ABDGHEICFJ, GDHBEIAFJC
Making tree...  Done
Report tree postorder : GHDIEBJFCA
Report tree level order : ABCDEFGHIJ

Input : ABDGHEICFJ, ABDGHEICFJ
Making tree...  Unaccepted input
Report tree postorder :
Report tree level order:
```

2. Complete `LCA.java`

    a. Create a `ExtTNode` class, which in addition to maintaining the node value (of type `int`) and references to the left and right subtrees, adds a reference to the parent of the node. This allows movement up the tree as well as down the tree.

    b. `ExtTNode LCA(ExtTNode n1, ExtTNode n2)`

        - returns the lowest common ancestor between the two nodes

    c. `void PrintCommonPath(ExtTNode n1, ExtTNode n2)`

        - prints the path common to the two paths from the given nodes **n1** and **n2** to the root, e.g. ‘`4->3->1`’
        - prints ‘`No Common Path`’ if either of the nodes are null

Note

- The skeleton code is a minimum (please do not change given class, method, and data member names). You may create more methods, data members, and classes as needed.
- In a given tree, there won’t be repeating node values.
- Unmentioned exceptions/errors do not need to be handled.

Example (running LCA.java)

```
LCA : 2
CommonPath : 2->1
```

Assessment:

- Copying others work will result in zero.
- If your program doesn't compile, then you will get a zero score.
- Grading will be done in the Linux environment with Java 11. If you comment or use any Korean character, there will be an error during the process of compiling.

Submission:

- By eTL. Make sure you write your name and student number as a comment on top of `MyTree.java` and `LCA.java.`
- Compress your `MyTree.java` and `LCA.java` into a zip or tar file, and rename as "<studentid>.zip", e.g. "2019-11111.zip"
- Do NOT email for changing in the assignment after you have submitted.
- Late submissions will not be accepted