CS256 Advanced Programming – Project 2

This is a project for design algorithms for a linked based binary tree. Please place the required files in folder **XXX_Project2**, and submit **XXX_Project2.tar.gz**.

Problem

Design algorithms for the following operations for a linked binary tree *T*:

- **preorder_next(p):** Return the position visited after p in a preorder traversal of T (or None if p is the last node visited).
- **inorder_next(p):** Return the position visited after p in an inorder traversal of T (or None if p is the last node visited).
- **postorder_next(p):** Return the position visited after p in a postorder traversal of T (or None if p is the last node visited).
- **delete_subtree(p):** Remove the entire subtree rooted at position p, making sure to maintain the count on the **size** of the tree.

Requirements:

(1) Create another python file **extend_linked_binary_tree.py**

(2) In this new file, define **ExtendedLinkedBinaryTree** as a **subclass** class of LinkedBinaryTree.

(3) **Implement** the four methods as described, add **comments** for each method and control block

   a. preorder_next(p)
   b. inorder_next(p)
   c. postorder_next(p)
   d. delete_subtree(p)

   Important: do not directly use the following methods when you implement preorder_next(p), inorder_next(p), postorder_next(p):
   preorder(), _subtree_preorder(p), inorder(), _subtree_preorder(p),
   postorder() , _subtree_preorder(p) .

(4) In the **test code**, you should test each method.

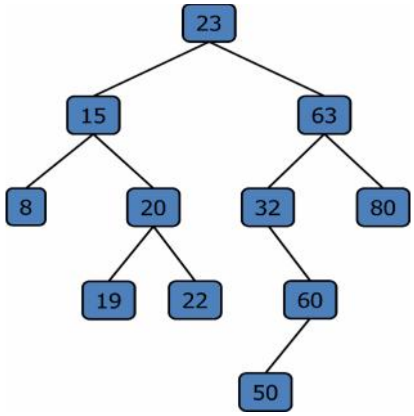
Make sure that:

- You test code provides **statement coverage** for the methods (refer to chapter 2: testing)
- **Catch** possible **TypeError** and **ValueError** Exception, and print corresponding error message.
- Your program running result is **readable**:
  You can print some info before you print an element or a sequence of elements.
  You could add additional methods to support your test.

For comparison, you may call methods in the following sequence:

   a. Build a linked binary tree
   b. Associate p with the position of a node in the binary tree, print the element in position p
   c. Call preorder(), and print the elements store in the tree in preorder traversal
   d. Call preorder_next(p), and print the element in the return position
   e. Call inorder(), and print the elements store in the tree in preorder traversal
   f. Call inorder_next(p), and print the element in the return position
   g. Call postorder(), and print the elements store in the tree in preorder traversal
   h. Call postorder_next(p), and print the element in the return position
   i. Call delete_subtree(p), and then call inorder() and print the elements in the remaining tree

For example, given the following binary tree:



Suppose p is position for node 32.
Step (c), using the preorder traversal, could print:
Preorder: 23 15 8 20 19 22 63 32 60 50 80
Step (d) prints: **Preorder_next(32): 60**
Step (e), using the inorder traversal, could print:
Inorder: 8 15 19 20 22 23 32 50 60 63 80
Step (f) prints: **Inorder_next(32): 50**
Step (g), using the postorder traversal, could print:
Postorder: 8 19 22 20 15 50 60 32 80 63 23
Step (h) prints: **Postorder_next(32): 80**

Step (i) delete_subtree(p) removes the left sub-tree of node 63. Then inorder traversal will print:
Inorder after delete subtree rooted at 32: 8 15 19 20 22 23 63 80

Submission:

- **All Python files** that are needed for run your program **(55 points)**
  - **linked_queue.py** (use the version you downloaded from Moodle in Chapter 7)
  - **tree.py** (download from the textbook's website)
  - **binary_tree.py** (download from the textbook's website)
  - **linked_binary_tree.py** (download from the textbook's website)
  - **extend_linked_binary_tree.py**
    - In this file, class ExtendLinkedBinaryTree is defined as sub-class of LinkedBinaryTree
    - (40 points) The four required methods are implemented
      - Code (30 points)
      - Well documented code and comments (10 points)
        - Refer to Chapter 2 : Coding Style and Documentation
    - (15 points) Provide test code in the control block if __name__='__main__':
- A **typescript** of running your program **(5 points)**
  - If you forget how to save the shell window content to a typescript, refer to Assign4 - Exercise 4.1 - Part 6.
- A **design document** of your class ExtendLinkedBinaryTree (PDF file) **(40 points)**
  - Part 1 (5 points): Draw the **hierarchy architecture** of the classes you used in this project (refer to Figure 2.4)
  - Part 2 (20 points): Draw the **flow chart** (refer to Figure 1.6) for each of the four methods
  - Part 3 (10 points): Draw the binary tree you used in your test code, list all test cases for each method.
  - Part 4 (5 points): State the **problem**(s) you encountered in this project.