LAB 10

Learning Outcomes

- Understand the linked list implementation of a binary tree
- Design and implement operations in the tree structure
- Apply recursion to traversals and other algorithms in a binary tree

Pre-Lab

- Create a new Java project called Lab10
- Download the files: BinaryTreeNode.java, LinkedBinaryTree.java, ArrayList.java, ArrayUnorderedList.java, ArrayIterator.java, BinaryTreeADT.java, ListADT.java, ElementNotFoundException.java, EmptyCollectionException.java, LinkedQueue.java, QueueADT.java, LinearNode.java, TestBinaryTree.java, and UnorderedListADT.java
- Save these downloaded files into the Lab10 src folder

Exercise 1 – Completing the BinaryTreeNode class

1. Open BinaryTreeNode.java. Note that it is currently very short.
2. Add the following getter methods: `getData()`, `getLeft()`, and `getRight()`.
3. Add the following setter methods: `setData(newData)`, `setLeft(newLeft)`, and `setRight(newRight)`.

Exercise 2 – Completing the LinkedBinaryTree class

1. Open LinkedBinaryTree.java. Note that there are several errors throughout this file, although not as many now as there were before you completed the above exercise (since the getters and setters in BinaryTreeNode are called many times from this class).
2. Complete the method `getDataRoot()` to return the data item stored in the root of the tree.
3. Complete the method `isEmpty()` to return true if the tree is empty and false otherwise
4. Complete the method `size(BinaryTreeNode<T> r)` to return the number of nodes in the subtree with root r. This method must be recursive. Please try to implement this method on your own. If you get stuck, then read this hint.

5. Complete the method `contains(BinaryTreeNode<T> r, T targetElement)` to return true if the given targetElement is stored in any of the nodes in the subtree with root r. This method must be recursive. Please try to implement this method on your own. If you get stuck, then read this hint.
6. Complete the methods `iteratorPreOrder()` and `preorder()` to work together to construct an iterator that recursively traverses the tree in pre-order (see note below).
7. Complete the methods `iteratorPostOrder()` and `postorder()` to work together to construct an iterator that recursively traverses the tree in post-order (see note below).
8. If you need help with any of the preorder or postorder iterator methods, read class TestBinaryTree.java to learn how to use the iterator. You can also model your code for these methods on the existing `iteratorInOrder()` and `inorder()` methods. The main change you will need to make is where the node is being visited.
9. Complete the method `iteratorLevelOrder()` to return an iterator containing the data stored in the nodes of the tree in level order. This method is not recursive. Use class LinkedQueue.java as the auxiliary data structure needed to perform a level order traversal of the tree. Review the lecture notes on tree traversal. Note that in the queue you will store tree nodes, so what should be the value of the generic type for the declaration of the queue? (`LinkedQueue< ??? > queue;`)
10. Open TestBinaryTree.java and run it. All 6 tests must pass to indicate that your code was entered correctly. If any of these tests are failing, use the Debugger and/or print lines to find the errors and fix them. Start by reviewing the BinaryTreeNode to check if you accidentally assigned a value to an incorrect variable.

Submission

When you have completed the lab, navigate to the weekly module page on OWL and click the Lab link (where you found this document). Make sure you are in the page for the correct lab. Upload the files listed below and remember to hit Save and Submit. Check that your submission went through and look for an automatic OWL email to verify that it was submitted successfully.

Rules

- Please only submit the files specified below. Do not attach other files even if they were part of the lab.
- Do not ZIP or use any other form of compressed file for your files. Attach them individually.
- Submit the lab on time. Late submissions will receive a penalty.
- Forgetting to hit "Submit" is not a valid excuse for submitting late.
- Submitting the files in an incorrect submission page will receive a penalty.

- You may re-submit code if your previous submission was not complete or correct, however, re-submissions after the regular lab deadline will receive a penalty.

Files to submit

- BinaryTreeNode.java
- LinkedBinaryTree.java