CSCI 2110 Data Structures and Algorithms
Laboratory No. 6
Week of October 23rd, 2017

In this lab you will implement methods for binary trees. Most of the methods are simple recursive methods.

Marking Scheme
Each exercise carries 10 points. Your final score will be scaled down to a value out of 10.
Working code, Outputs included, Efficient, Good basic comments included: 10/10
No comments: subtract one point
Unnecessarily inefficient: subtract one point
No outputs: subtract two points
Code not working: subtract up to six points depending upon how many methods are incorrect.

Error checking*: Unless otherwise specified, you may assume that the user enters the correct data types and the correct number of input entries, that is, you need not check for errors on input.*

Submission*: All submissions are through Brightspace. Log on dal.ca/brightspace using your Dal NetId. Submissions are pretty straightforward. Instructions will be also be given in the first lab.*
Deadline for submission: Sunday, October 29th, 2017 at 11.55 p.m. (five minutes before midnight).

NOTE: Do not hardcode any of the input values in your source code. Prompt the user to enter the values.

Download the following files from the course website (alongside the Lab6 link):
BinaryTree.java, BinaryTreeDemo.java
We discussed these in the lectures. Study and understand the methods.

Step 1: Add a recursive method to BinaryTree.java to find the number of nodes in a binary tree.

It is easy to design this recursive method. The solution is given below:
if the binary tree is empty, then the number of nodes is zero,
otherwise, it is equal to one plus number of nodes in the left subtree plus number of nodes in the right subtree.

Test the method by making appropriate changes to BinaryTreeDemo.java for at least three different inputs.

Now design the following:

Step 2: Add a recursive method to BinaryTree.java to find the height of a binary tree. Test the method for at least three different inputs.

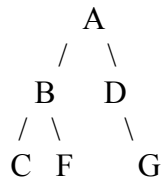
Step 3: A binary tree is height balanced if, for every node in the tree, the height of its left subtree differs from the height of its right subtree by no more than one. In other words, either

the left and the right subtrees are of the same height, or the left is one higher than the right, or the right is one higher than the left.

Add a recursive method that determines whether a specified binary tree is height balanced. Test the method for at least three different inputs.

Step 4: Add a static method
public static<T> void levelOrder(BinaryTree<T> tree)
that would perform a level order traversal of the specified tree.

Note: A level order traversal traverses the tree level by level, from left to right, starting at the root. For example, for the following tree:



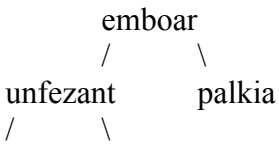
the level order traversal displays A B D C F G

Hint: This can be done iteratively. Use an ArrayList of type BinaryTree. Initially put the root of the tree in it. Remove it and put its children. Repeat the procedure until the ArrayList is empty.
Test the method for at least three different inputs.

Step 5: This is a generalization of the BinaryTreeDemo.java program. Write a program called BinaryTreeDemo1.java that reads a number of words from the keyboard and create a binary tree by putting the first word in the root and alternately attaching successive words as the left or the right child nodes. Then print the height of the tree, the number of nodes in the tree and the inorder, preorder, postorder and level order traversals.
For example, here's a sample dialog:

Enter name or done: emboar
Enter name or done: unfezant
Enter name or done: palkia
Enter name or done: serperior
Enter name or done: samurott
Enter name or done: done

The tree is:



serperior samurott

Height of the tree is: 2
Number of nodes in the tree is: 5
Inorder: serperior unfezant samurott emboar palkia
Preorder: emboar unfezant serperior samurott palkia
Postorder: serperior samurott unfezant palkia emboar
Level order: emboar unfezant palkia serperior samurott

Note: You must submit a zip file containing the source codes (BinaryTree.java, BinaryTreeDemo.java and BinaryTreeDemo.java ONLY) and sample outputs (in text format). Do not submit unreadable .class files.