CSCI 2110 Data Structures and Algorithms
Fall 2023
Assignment No. 4

Date Given: Monday, October 30, 2023
Due (on Brightspace): Monday, November 13, 2023, 11.59 PM

The objective of this assignment is to help you get familiar with the binary tree data structure. You will need the following files to complete your work. They have been provided to you along with this document. Download them.

BinaryTree.java (Generic BinaryTree Class)
BinaryTreeDemo.java (A class demonstrating the methods of the BinaryTree class)

Exercise 1 (Binary Tree Methods)

For this exercise, you will complete four methods listed as TODO in the BinaryTree.java file.

1. Complete the recursive method called **nodes** in the BinaryTree.java file. This method should return the number of nodes in a binary tree.

To determine the number of nodes in a binary tree you can apply the following rules:

- if the binary tree is empty, then the number of nodes is zero; otherwise, the number of nodes is equal to one plus number of nodes in the left subtree plus number of nodes in the right subtree.

2. Complete the recursive method called **height** in the BinaryTree.java file. This method should return the height of a binary tree.

To determine the height of a binary tree you can apply the following rules:

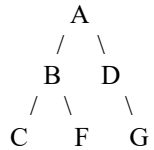
- if the binary tree is empty, then the height of the binary tree is -1; otherwise, the height is equal to 1 plus the height of either the left subtree or the right subtree, whichever is greater.

3. Complete the recursive isBalanced method in the BinaryTree.java file. This method should return a Boolean reflecting whether or not a binary tree is height balanced.

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.

4. Complete the levelorder method in the BinaryTree.java file. This static, dynamic method should perform a level order traversal of a binary tree passed to it as an argument.

A level order traversal goes level by level, from left to right, starting at the root. The level order traversal of the following binary tree:



is A B D C F G

To accomplish this, you will implement a simple breadth first search. One way to implement breadth first search is to use an ArrayList of type BinaryTree. First add the root node to the ArrayList. Traverse the tree

by removing the first element from the ArrayList, printing, and adding the children of the node removed to the ArrayList. Repeat until your ArrayList is empty.

Now make appropriate changes in the BinaryTreeDemo.java file and test the methods you have written with at least three different sets of inputs.

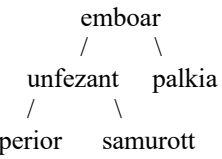
Exercise 2

For this exercise you will develop a program that can build a binary tree from user input. Create an Exercise2 class (Exercise2.java). You may reuse and expand upon code provided in the BinaryTreeDemo.java file, but your program must **accept input from a user** (an arbitrary number of Strings) and build a binary tree with String data from that input.

The most straightforward approach is to create single node binary trees from each String captured, storing them in an ArrayList of type BinaryTree. You can then quickly build a larger binary tree by assigning the first tree in your ArrayList to be the root node, and attaching subsequent elements to that root.

After building a binary tree, your program should display the height of the tree, the number of nodes in the tree, the inorder, preorder, postorder and level order traversals. You may reuse the test code from the BinaryTreeDemo.java file to this end.

Note: You are building a complete binary tree. It will, by definition, be height balanced. If the Strings provided by the user are emboar, unfezant, palkia, serperior, and samurott, the resulting tree will look like this:



A sample run of your Exercise2 program should look like this:

```
Enter data for binary tree:
emboar unfezant palkia superior 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
```

Test your program with at least three different sets of inputs.

What to submit:
Submit one ZIP file containing all source codes (files with .java suffixes), inputs and outputs of Exercise1 and Exercise2.
You MUST SUBMIT .java files that are readable by your TAs. If you submit files that are unreadable such as .class, you will lose points. Please additionally comment out package specifiers.
Late Submission Penalty: The assignment is due on Monday (November 13) at 11.59 PM. Late submissions up to 5 hours (4.59 AM on Tuesday) will be accepted without penalty. After that, there will be a 10% late penalty per day on the mark obtained. For example, if you submit the assignment on Tuesday at 12 noon and your score is 8/10, it will be reduced to 7.2/10. Submissions past five days after the grace submission time will not be accepted.