CSI 33 Fall 2024 Review PROBLEMS (Ojakian)

1. Recall that we can build a binary tree using the following class:

   ```
   class TreeNode:
       def __init__(self, data, left, right):
           self.item = data
           self.left = left
           self.right = right
   ```

   For example, we can build a tree with a root and 3 other nodes as follows:

   ```
   root = TreeNode(5, None, None)
   n1 = TreeNode(4, None, None)
   n2 = TreeNode(3, None, None)
   n3 = TreeNode(7, None, None)
   root.left = n1
   root.right = n2
   n1.right = n3
   ```

   (a) Draw the tree in the above example. Is the tree full? It is complete? Is it any of the data structures we have discussed in the course (heap, BST, etc), and why or why not?

   (b) Write the definition of a function `sumPath` which takes two inputs (a binary string *P* and the root node of a tree). Start at the root node, and move down the tree according to the directions in string *P* (i.e. a '0' means move left, and a '1' means move right), summing all the `item` values. Then return this value. For example, in the above tree `sumPath('01', root)` should return 16 since $5+4+7=16$.

   You can assume that the tree at least has a root node and that the string *P* describes a valid way to move through the tree. Your function must work for any binary tree, not just the example above.

   (c) Write the definition of a function `countNodes` which takes one input (the root node of a tree) and returns the number of nodes in the tree. For example, in the above tree `countNodes(root)` should return 3 since there are 3 nodes in the tree

2. Do the `countNodes` function from above but for our C++ ListNode.

3. Consider the binary tree data types: BST, Min Heap, Max Heap. Give 3 examples, where each example has 4 nodes and is exactly one the last data types.

4. Give an example of a BST which is not an AVL tree. What is the fewest number of nodes for which this is possible?

5. Give an example of an AVL tree which is full, one which is complete but not full, and one which is not even complete.

6. Make a diagram of a Linked List with the elements 5, 3, 0. Code it up from scratch just using the ListNode class - do this in Python and in C++ (in C++ we use pointers). Try the following, making a diagram, and coding it up in Python and C++. Try to do it by just referring to the "head" pointer and using the links. When deleting in C++, make sure you avoid memory leaks.

   (a) Make a diagram to show ALL the steps for inserting a 7 between the 3 and the 0.

(b) Make a diagram to show ALL the steps for inserting a 9 at the end of the list.

(c) Make a diagram to show ALL the steps for inserting a 2 at the beginning of the list.

(d) Show ALL the steps for deleting the third node.

(e) Show ALL the steps for deleting the last node.

(f) Show ALL the steps for deleting the first node.

7. Write a C++ function `minL`(head) which takes the ListNode pointer head as input and returns the minimum integer in the linked list.

8. Write a Python function which takes in a Queue and a Stack as input and checks if the second from the front element of the Queue is equal to the second from the top element of the Stack. The Stack and Queue should be unchanged after the function call. And you can only use the 4 standard methods for Stack (push, etc) and the 4 standard ones for Queue (enqueue, etc).

9. Do a Theta analysis of your last program.

10. Write a Python function which takes in a list of Queues and returns true if they all have the same front value, and false otherwise.

11. Write a Python function which takes in a Queue and a Stack and returns true if their elements are the same, where we read a Queue front to back, and a Stack from top to bottom. Neither should be changed after the function call.