In each Section, choose only 1 question

Section A

a. Given two sorted stacks stack1 and stack2, implement a function mergeStacks that merges the content of stack1 and stack2 in a new stack result_stack in such a way that the result stack is also sorted (either descendinly or ascendingly).

b. Given two stacks stack1 and stack2, write a function isSimilar that checks whether the two stack are exactly similar and returns true if it is the case; otherwise, it return false.

Section B

a. Assume a linked list that holds integers. Suppose that you want to count how many positive values are on the list. Write a function countPositive that returns the count of values.

b. Assume a linked list that holds at least 2 integers. Suppose that you want to insert a new node in the end of the list ONLY if its value is greater than the value of the last node of the list. Write a function insertWithCondition that performs the insertions.

Section C

a. Write a method trim that could be added to the IntTree class from lecture and section. The method accepts minimum/maximum integers as parameters and removes from the tree any elements that are not in that range, inclusive. For this method, assume that your tree is a binary search tree (BST) and that its elements are in valid BST order. Your method should maintain the BST ordering property of the tree.

b. Write a method nodesAtLevels that could be added to the IntTree class from lecture and section. The method accepts minimum and maximum integers as parameters and returns a count of how many elements exist in the tree at those levels, inclusive. Recall that the root of a tree is at level 1, its children are at level 2, their children at level 3, and so on.

Section D

a. Write a program to implement a sort technique that works on the principle of divide and conquer strategy.

b. Write a program to implement partition-exchange sort.