Lab 2

Release: 19 Oct 2020 (Mon, Week 10)
Due: Online submission in eLearn by Sunday, 25 Oct 2020, 6pm

Instructions:

1. This is an individual lab.
2. If need be, you can discuss the general approach to solve the questions within your assigned study groups but you should not be discussing or sharing your codes with your team members or anyone, including in Piazza. Piazza should only be used for the purpose of clarifying the requirements (if any).
3. You are allowed to create additional methods in the classes.
4. Please ensure that all your codes can be compiled.
5. Please include meaningful comments at the block of codes that you have written to explain your logic.
6. We may use additional test cases other than the ones provided to test the submissions.
7. Multiple submissions are allowed and we will take the submission closest to the deadline as the final submission.
8. Please zip up your submission as named the zip file as <StudentName>.zip.
9. For efficiency purpose, you can just focus on time complexity.
10. The deadline is final, late submissions will not be accepted.

Grading Guidelines : (5% of final grade)

Grade	Expectations
A and above	Complete and working solutions with clear and concise comments to support the implementation. Implementations are efficient with the appropriate use of data structures, algorithms and considerations.
B and above	Complete and working solutions with clear and concise comments to support the implementation
Below B	Incomplete submissions, or some codes cannot be compiled.

Q1 – Find integers in a linked list that sum up to a given value

Given a Integer LinkedList, the objective is to find the specified number of integers within the linked list that sum up to a given value.

- Write a method named **findIntegers** that takes in 3 parameters (LinkedList<Integer> input, int numOfIntegers, int value) and returns a string containing the integers that sum up to the value
- E.g. for LinkedList input [1, 2, 3, 4], findIntegers(input, 1, 4) should return "(4)".
- E.g. for LinkedList input [1, 2, 3, 4], findIntegers(input, 2, 4) should return "(1,3)".
- E.g. for LinkedList input [3, 4, 1, 2], findIntegers(input, 2, 5), should return "(3,2), (4,1)"
- The order of the output doesn't matter
    - "(1,3)" or "(3,1)" is acceptable
    - "(3,2), (4,1)" or "(2,3), (1,4)" or "(4,1), (3,2)," or "(1,4), (2,3)," is acceptable
- The LinkedList input may be unsorted and values are unique.
- You can assume it will only accept Integer type LinkedList
- You can create additional classes as part of the solution

Input : [1, 2, 3, 4, 5, 6]

Find 1 integer(s) that add up to 7
Expected Result : Invalid

Find 1 integer(s) that add up to 3
Expected Result : (3)

Find 2 integer(s) that add up to 4
Expected Result : (1,3)

Find 3 integer(s) that add up to 6
Expected Result : (1,2,3)

Find 2 integer(s) that add up to 7
Expected Result : (1,6), (2,5), (3,4)

****

Input : [3, 1, 2, 4]

Find 2 integer(s) that add up to 4
Expected Result : (1,3)

Q2 – Verify traversals of a binary search tree

Given 3 traversals, the objective is to verify if the 3 traversals is a valid combination of a binary search tree. Compile and run Q2Test.java to test your implementation.

- Write a method named **verify** that takes in 3 strings parameter (String traversal1, String traversal2, String traversal3) containing the traversal of a binary search tree.
- The input parameters could be of any sequence e.g. traversal1 could be Inorder, Preorder, Postorder
- The nodes in the sequence are separated by "-" and contains integers only e.g. (1-2-3)
- The nodes are unique in a traversal. e.g. (there won't be 1-2-2-3)
- If the 3 traversals are valid, the output prints the traversal orders of the respective strings. Otherwise, prints "Invalid traversals"
- You can create additional classes as part of the solution

Traversal 1 : 1-2
Traversal 2 : 2-1-3
Traversal 3 : 1-3-2
Expected Result : Invalid traversals
Actual Result : Invalid traversals

Traversal 1 : 1-2-3
Traversal 2 : 2-1-3
Traversal 3 : 1-3-2
Expected Result : Traversal 1 - Inorder, Traversal 2 - Preorder, Traversal 3 - Postorder
Actual Result : Traversal 1 - Inorder, Traversal 2 - Preorder, Traversal 3 - Postorder

Traversal 1 : 1-2-3
Traversal 2 : 3-2-1
Traversal 3 : 2-3-1
Expected Result : Invalid traversals
Actual Result : Invalid traversals

Traversal 1 : 3-1-2-5-4
Traversal 2 : 2-1-4-5-3
Traversal 3 : 1-2-3-4-5
Expected Result : Traversal 1 - Preorder, Traversal 2 - Postorder, Traversal 3 - Inorder
Actual Result : Traversal 1 - Preorder, Traversal 2 - Postorder, Traversal 3 - Inorder

Traversal 1 : 10-20-30-40-50
Traversal 2 : 20-10-30-40-50
Traversal 3 : 10-50-40-30-20
Expected Result : Traversal 1 - Inorder, Traversal 2 - Preorder, Traversal 3 - Postorder
Actual Result : Traversal 1 - Inorder, Traversal 2 - Preorder, Traversal 3 - Postorder