Singly Linked List [CO3]

[Each method carries 5 marks]

Instructions for students:

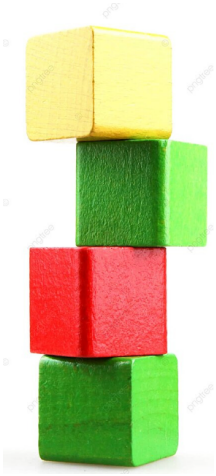
- Complete the following methods on Singly Linked List.
- You may use any language to complete the tasks.
- All your methods must be written in one single .java or .py or .pynb file. DO NOT CREATE separate files for each task.
- If you are using JAVA, you must include the main method as well which should test your other methods and print the outputs according to the tasks.
- If you are using PYTHON, then follow the coding templates shared in this folder.

NOTE:

- **YOU CANNOT USE ANY OTHER DATA STRUCTURE OTHER THAN LINKED LIST**
- **YOUR CODE SHOULD WORK FOR ANY VALID INPUTS. [Make changes to the Sample Inputs and check whether your program works correctly]**
- **LOOK OUT FOR 0th NODE Condition**

1. Building Blocks

Your twin and you are under an experiment where the amount of thinking similarities you two have is being observed. As per the experiment, you are given the same number of building blocks of different colors and are told to make a building using those blocks in two different rooms.



After the buildings are finished, the observers check whether the two buildings are the same based on the block colors. Now, you are the tech guy of that team and you are instructed to write a program that will output “Similar” or “Not Similar” given the two buildings. For fun, you decided to represent those buildings as a linked list!

NB: Red means a red block
Blue means a blue block
Yellow means a yellow block
Green means a green block.

Sample Input	Sample Output
building_1 = Red→Green→Yellow→Red→Blue→Green→None building_2 = Red→Green→Yellow→Red→Blue→Green→None	Similar
building_1 = Red→Green→Yellow→Red→Yellow→Green→None building_2 = Red→Green→Yellow→Red→Blue→Green→None	Not Similar
building_1 = Red→Green→Yellow→Red→Blue→Green→None building_2 = Red→Green→Yellow→Red→Blue→Green→Blue→None	Not Similar

2. Remove Compartment

Sheldon is a train maniac. He loves attaching each compartment of a train to the next with a linking chain as his hobby. Now he is removing the nth compartment from the end of the train. Can you model the scenario by writing a method that takes the compartment sequence and a number; the method returns the changed (or unchanged) train compartment sequence.

Constraint:

a. You **cannot** create any new linked list. You have to change the given one.

Sample Input	**Sample Output**
10→15→34→41→56→72 2	10→15→34→41→72
10→15→34→41→56→72 7	10→15→34→41→56→72
10→15→34→41→56→72 6	15→34→41→56→72

3. Assemble Conga Line

Have you ever heard the term conga line? Basically, it's a carnival dance where the dancers form a long line. Everyone holds the waist of the person in front of them and their waists are held in turn by the person to their rear, excepting only those in the front and the back. It kind of looks like this-

By now, you can quite understand the suitable data structure to represent a conga line. Now you are the choreographer of the Conga Dance in a Summer Festival. You wish to arrange the conga line **ascending** age wise and tell the participants to stand in a line likewise. Now as technical you are, can you write a method that will take the conga line and return True if everyone stands according to your instruction. Otherwise returns False.

Sample Input 10→15→34→41→56→72 Sample Input 10→15→44→41→56→72	Sample Output True Sample Output False

4. Word Decoder

Suppose, you have been hired as an cyber security expert in an organization. A mysterious code letter has been discovered by your team, your task is to decode the letter. After lots of research you found a pattern to solve the problem. Problem details are given below:

- For each encoded word a linked list will be given, where each node will carry one letter as an element.
- For the decoded word, the letters are at the multiples of $(13\ \%\ length\ of\ linkedlist)$ position **[0 indexed].** For example, if the length of the given linked list is 10, then 13 % 10 = 3 and the letters are at positions which are multiples of 3 (i.e. at position 3, 6, 9)
- However, the letters are stored in the given linked list in opposite order. Thus the decoded word has to be reversed.
- After decoding, your program should return a dummy headed singly linked list containing the decoded word.

Sample Input	Sample Output
B→M→D→T→N→O→A→P→S→C	None → C→A→T **Explanation:** Length of the list= 10 & 13%10=3 The letters are T, A, C at 3, 6, 9 positions respectively **[0 indexed].** The letters are reversed and the resulting linked list is None → C→A→T
Z→O→T→N→X	None → N **Explanation:** Length of the list= 5 & 13%5=3. The letter is N at position 3 **[0 indexed].** The letters are reversed and the resulting linked list is None → N

5. Alternate Merge

You are given two singly non-dummy headed linked lists. Your task is to write a function **alternate_merge(head1, head2)** that takes the heads of the two linked lists and returns the head of a modified linked list with all the elements of the two lists in alternate order. It is guaranteed that alternate placement is always possible. Your resulting linked list will always start with the head of linked list 1.

Input	Output
List1: 1 → 2 → 6 → 8 → 11 → None List2: 5 → 7 → 3 → 9 → 4 → None	1 → 5 → 2 → 7 → 6 → 3 → 8 → 9 → 11 → 4 → None
List1: 5 → 3 → 2 → -4 → None List2: -4 → -6 → 1 → None	5 → -4 → 3 → -6 → 2 → 1 → -4 → None
List1: 4 → 2 → -2 → -4 → None List2: 8 → 6 → 5 → -3 → None	4 → 8 → 2 → 6 → -2 → 5 → -4 → -3 → None

6. Sum of Nodes

You are given a Linked List, LL1, and an array, dist. Write a method **sum_dist(list, arr)** that takes a Linked List and an array as parameters. This method sums the node elements in the linked list that are away from the head by the elements in the array and returns the sum. Assume the Node class has only elem and next variable. **No need to write Node class and driver code.**

Sample Input	Sample Output	Explanation
LL1 = 10--> 16 --> -5 --> 9 --> 3 --> 4 dist = [2, 0, 5, 2, 8] Function Call: print(sum_dist(LL1, dist))	4	Node Element away from the head at distance 2 = -5 Node Element away from the head at distance 0 = 10 Node Element away from the head at distance 5 = 4 Node Element away from the head at distance 8 = Doesn't Exist, Considered as 0 The sum is: -5+10+4+-5+0 = 4

Bonus Task: ID Generator

Write a function **idGenerator()** that will take three non-dummy headed linear singly linked lists containing only digits from 0-9 and **generate another singly linked list** with 8 digit student ID from it [The student ID contains only digits from 0 to 9].

The first linked list will have the first four digits of the student id in reverse order. The remaining four digits can be obtained by adding the elements of the second linked list **from the beginning** with the elements of the third linked list **from the beginning**.

If the summation is >= 10, then you have to mod the summation by 10 and insert the result in the output linked list. For example, in sample input 2, the last element of the second list is 7 and the last element of the third linked list is 8. So after adding them we get 7+8=15>10. So the digit we will insert as an element of the Student ID list, will be 15%10=5.

Sample Input 1:	**Sample Input 2:**
0 → 3 → 2 → 2	0 → 3 → 9 → 1
5 → 2 → 2 → 1	3 → 6 → 5 → 7
4 → 3 → 2 → 1	2 → 4 → 3 → 8
Sample output 1:	**Sample output 2:**
2 → 2 → 3 → 0 → 9 → 5 → 4 → 2	1 → 9 → 3 → 0 → 5 → 0 → 8 → 5