LAB 2 : Singly & Doubly LinkedList

[CO3]

Instructions for students:

- Complete the following methods on Singly & Doubly Linked List.
- You may use Java / Python to complete the tasks.
- DO NOT CREATE a separate folder for each task.
- If you are using **JAVA**, then follow the **Java template**.
- If you are using **PYTHON**, then follow the **Python template**.

NOTE:

- **YOU CANNOT USE ANY OTHER DATA STRUCTURE OTHER THAN THE LINKED LIST YOU'RE CREATING.**
- **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**

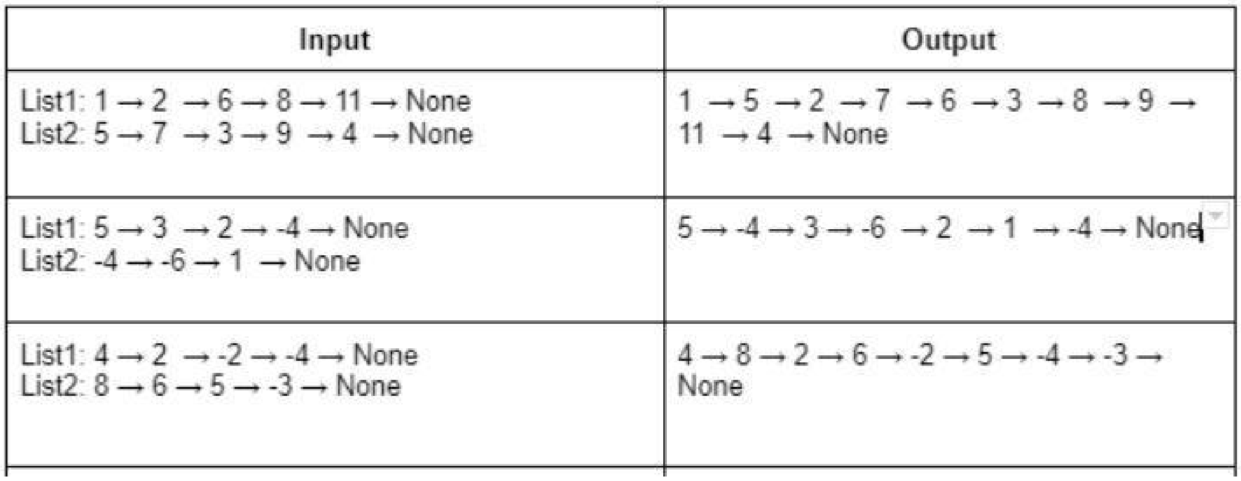
TOTAL MARKS: 40

Singly Linked List

[Each method carries 5 marks]

1. Alternate Merge [5 marks]

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

NOTE: You’re NOT ALLOWED to create a new linked list for this task

2. Word Decoder [5 marks]

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

3. ID Generator [5 marks]

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

Doubly Linked List

In this part of the lab, we will play with Dummy Headed Doubly Circular Linked List. If you want to read about this type of linked list then check this file.

```
Note: Even though we're playing with Dummy Headed Doubly Circular
Linked List, the methods you'll be writing simulates a different data
structure. Can you guess what that is? Hint: It's not taught yet and
it is something you face in front of the elevator everyday.
```

In the first part of this lab, you have to implement a waiting room management system in an emergency ward of a hospital. Your program will serve a patient on a **first-come-first-serve basis.**

Solve the above problem using a **Dummy Headed Doubly Circular Linked List.**

1. You need to have a **Patient** class so that you can create an instance of it (patient) by assigning id(integer), name (String), age (integer), and bloodGroup (String).
2. Write a **WRM** (waiting room management) class that will contain the below methods.
   a. **registerPatient(id, name, age, bloodgroup):** This method will register a patient into your system. The method will create a Patient type object with the information received as parameter. It means this method will add a patient-type object to your linked list. `[5 marks]`
   b. **servePatient():** This method calls a patient to provide hospital service to him/her. In this method, you need to ensure to serve the patient first who was registered first. `[5 marks]`
   c. **cancelAll():** This method cancels all appointments of the patients so that the doctor can go to lunch. `[2 marks]`
   d. **canDoctorGoHome():** This method returns true if no one is waiting, otherwise, returns false. `[1 mark]`
   e. **showAllPatient():** This method prints all ids of the waiting patients in sequential order. It means the patient who got registered first, will come first, and so on. `[2 marks]`
   f. **reverseTheLine():** This method reverses the patient line. It means the patient who got registered last, will come first, and so on. `[5 marks]`

3. Write a **Tester** code that will interact with users and take information about Patients. You will pass this information to **WRM** and create instances of **Patient** in **WRM** and call the methods of **WRM** class. You just need to ensure your Tester code has completed all the properties mentioned in the next point. `[5 marks]`

4. Tester Code Options:
    a. **registerPatient()** – print Success or Not
    b. **servePatient()** – print Name of Patient being Served
    c. **showAllPatients()** – print all patient in sequence to serve
    d. **canDoctorGoHome()** – return yes or no
    e. **cancelAll()** – print Success or Not
    f. **reverseTheLine() -** print Success or Not

Here's a video & A Simulation of WRM showing how the output of the Tester Code may look like. However, your output won't have to be exactly like this. As long as it fulfils the requirements above it should be fine.

Hints:

1. In your program, there won't be any class called **"Node".** Instead the **Patient class** will work as the Node class.
2. On the other hand, the **WRM** class will act as the Dummy Headed Doubly Circular Linked List which will contain all the other functions/methods.