**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**](https://drive.google.com/file/d/1v46ve_Y3vxYS8-Ow0AULVl-Q4QSAKxfo/view?usp=drive_link).
* If you are using **PYTHON**, then follow the [**Python template**](https://colab.research.google.com/drive/1OXQBbFpIsWGbsIpaIOQsqoaqQO84ZR4l).

## **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]**

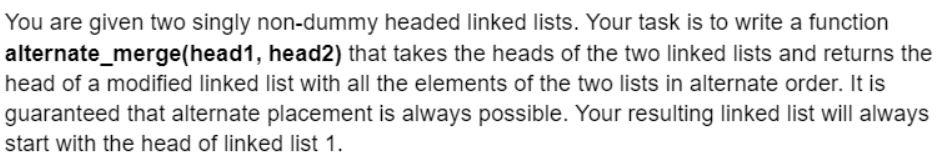
## **LOOK OUT FOR 0th NODE Condition**

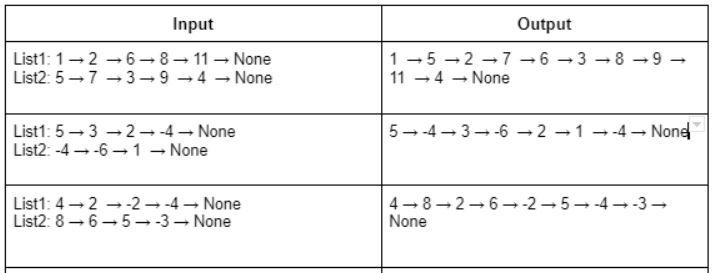
**TOTAL MARKS: 40**

# **Singly Linked List**

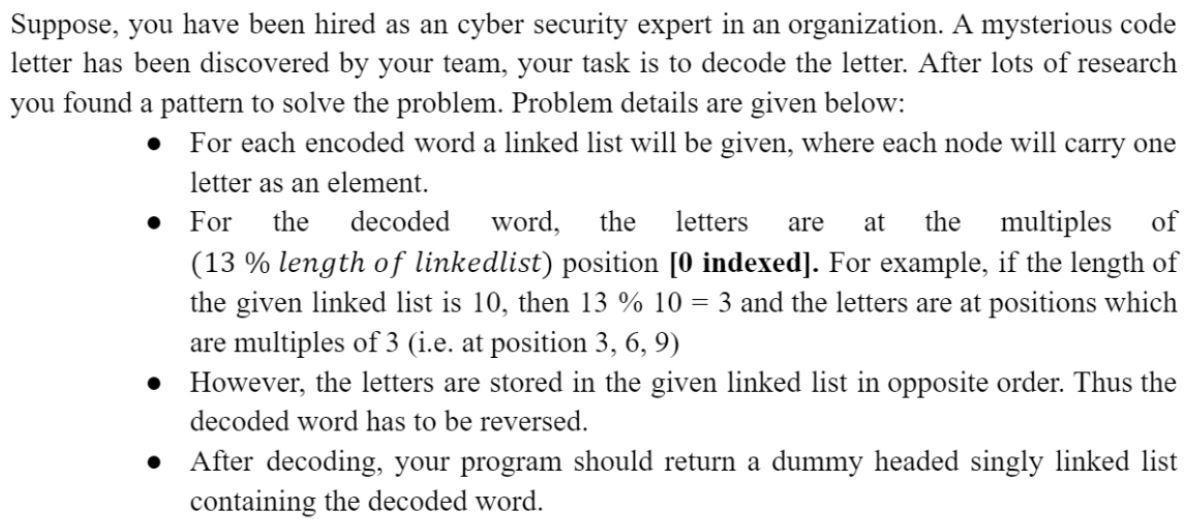
[Each method carries 5 marks]

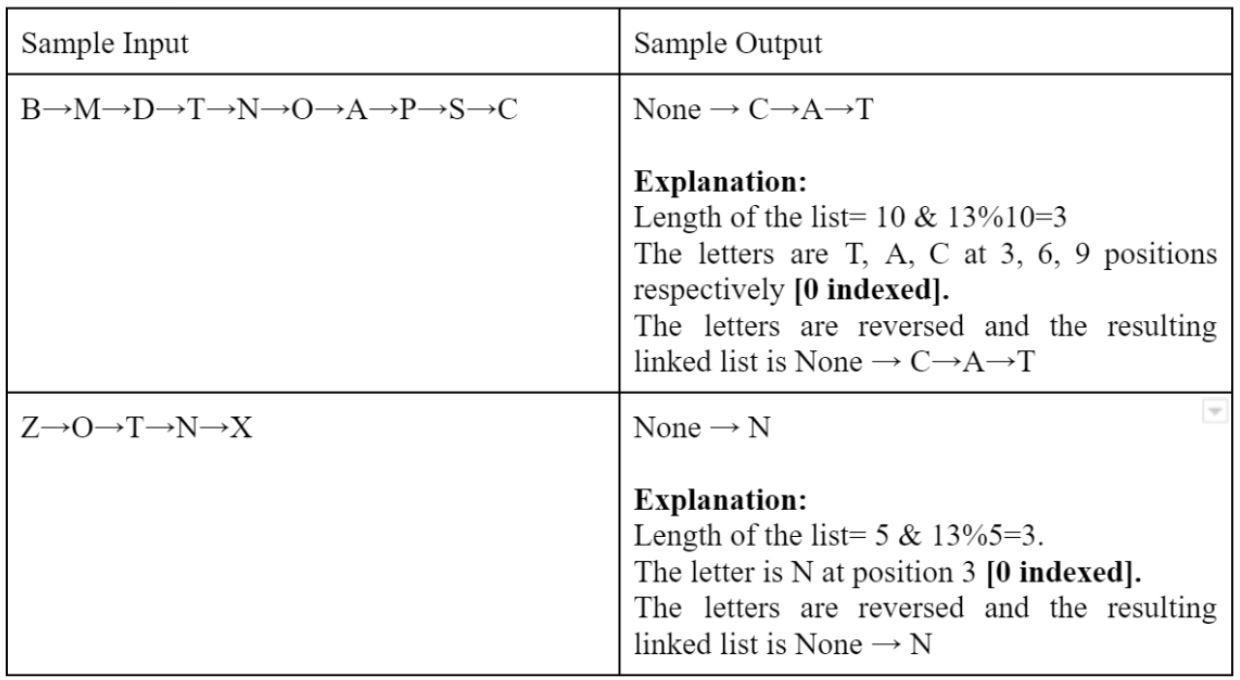
1. **Alternate Merge [5 marks]**

****

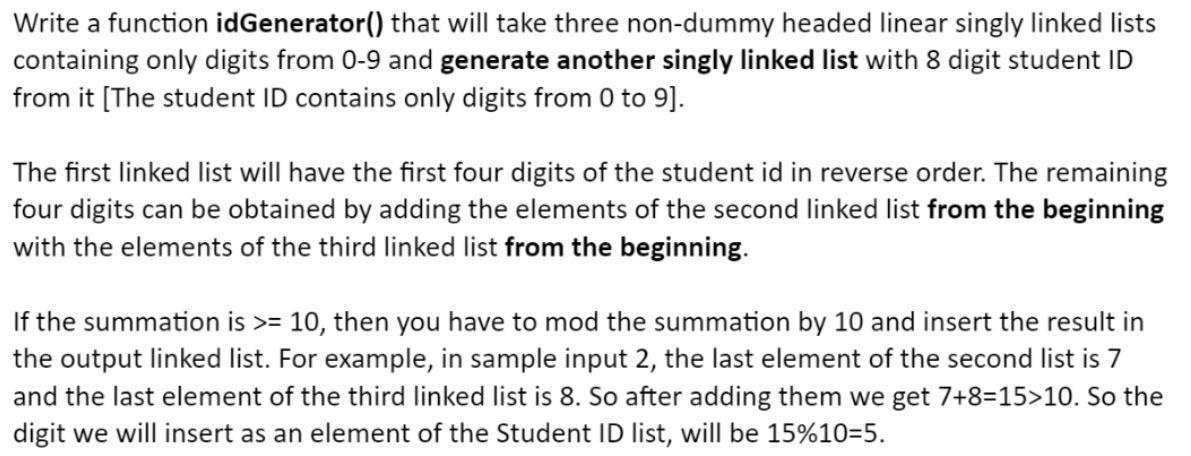
****

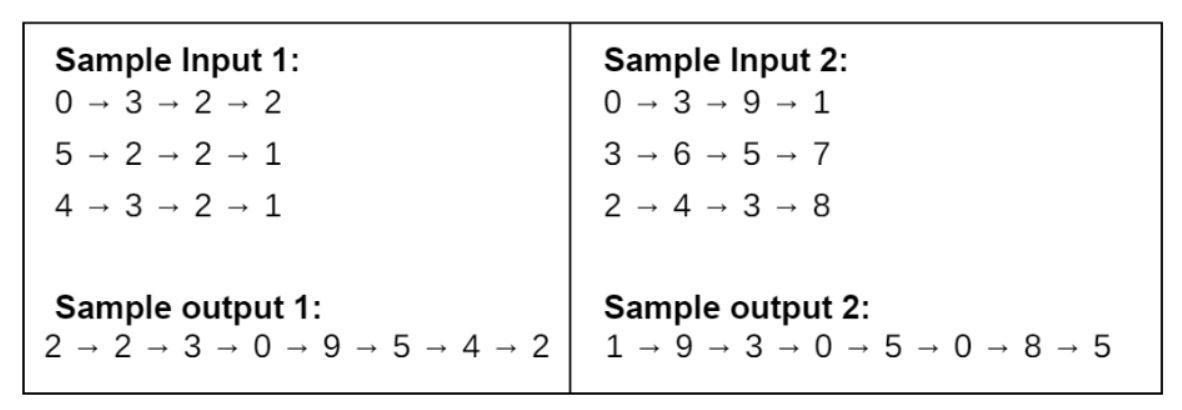
**NOTE: You’re NOT ALLOWED to create a new linked list for this task**

1. **Word Decoder [5 marks]**

****

1. **ID Generator [5 marks]**

****

****

# **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**](https://docs.google.com/document/d/1fm0pI9rLdGM1nme7jD_iA4ZrjRzBzKjg3uPh55IHyM4/edit?usp=sharing)**.**

| **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.
   1. **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]**
   2. **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]**
   3. **cancelAll():** This method cancels all appointments of the patients so that the doctor can go to lunch. **[2 marks]**
   4. **canDoctorGoHome():** This method returns true if no one is waiting, otherwise, returns false. **[1 mark]**
   5. **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]**
   6. **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:
   1. **registerPatient()** – print Success or Not
   2. **servePatient()** – print Name of Patient being Served
   3. **showAllPatients()** – print all patient in sequence to serve
   4. **canDoctorGoHome()** – return yes or no
   5. **cancelAll()** – print Success or Not
   6. **reverseTheLine()** - print Success or Not

| [**Here's a video**](https://drive.google.com/file/d/1DVSi7LQjJqE48qg-Y5t4Bsh9zGLE8oXk/view?usp=drive_link) **&** [**A Simulation of WRM**](https://drive.google.com/file/d/1ph4VJ-D_MCcnovquJeAkQqfVZ5B75_t9/view) 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.