## 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:**

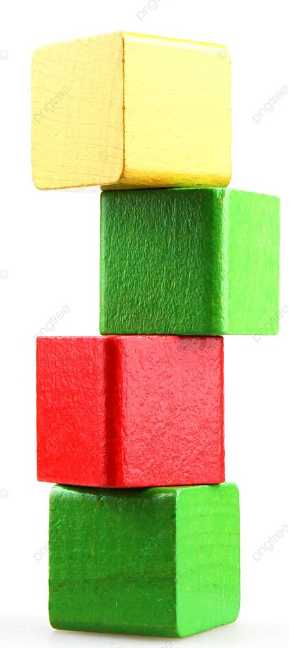
**● YOU CANNOT USE ANY OTHER DATA STRUCTURE OTHER THAN LINKED LIST**

**● 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**

### 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 |

### 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**:

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

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

### 

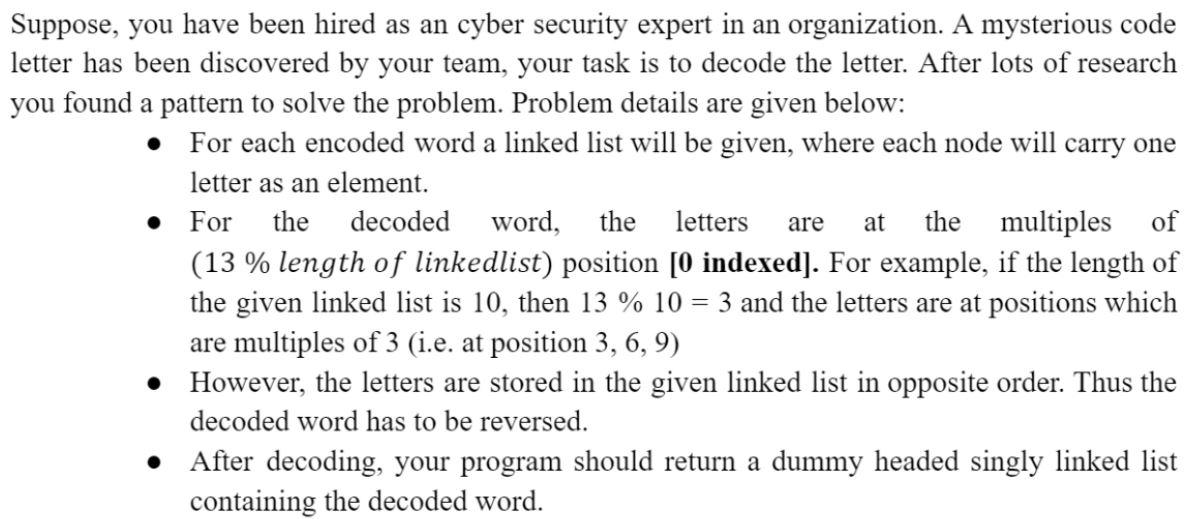
1. **Assemble Conga Line**

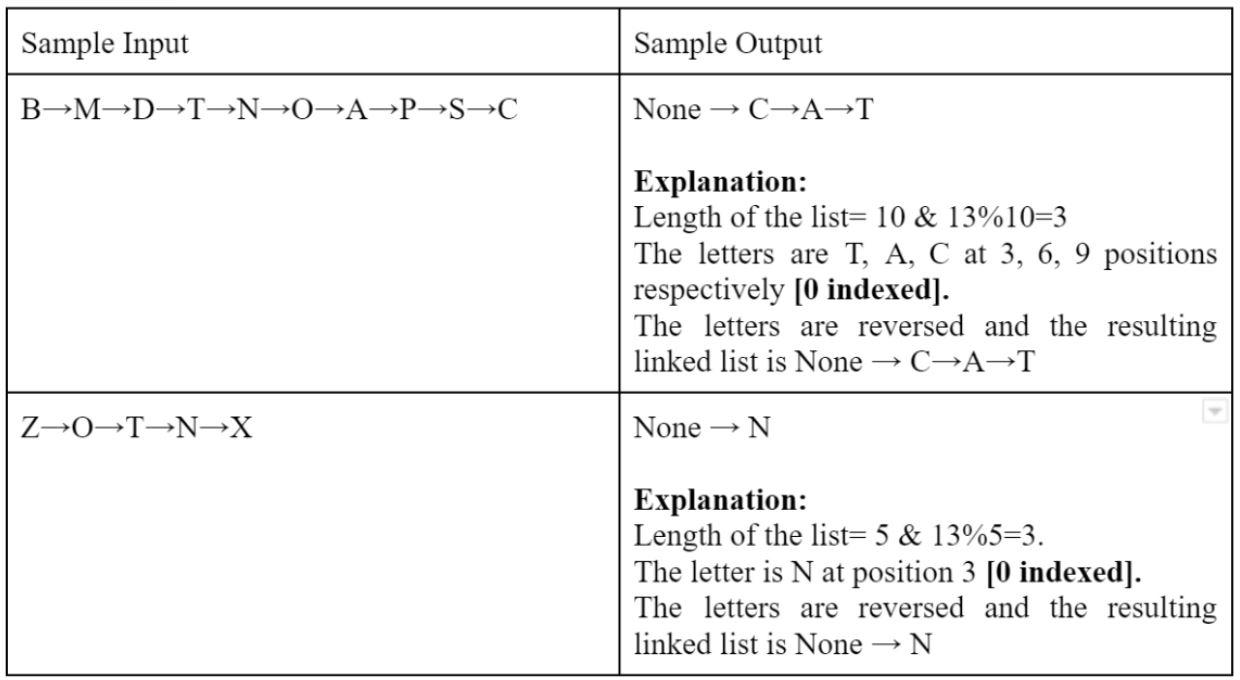
Have you ever heard the term [conga line](https://en.wikipedia.org/wiki/Conga_line)? Basically, it’s a carnival dance where the dancers form a long line. Everyone holds the waist of the person in front of them and their waists are held in turn by the person to their rear, excepting only those in the front and the back. It kind of looks like [this](https://www.pixtastock.com/illustration/11648015)-



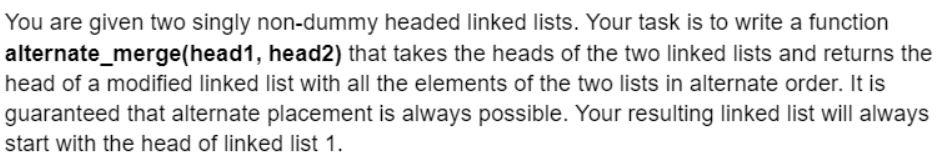
By now, you can quite understand the suitable data structure to represent a conga line. Now you are the choreographer of the Conga Dance in a Summer Festival. You wish to arrange the conga line **ascending** age wise and tell the participants to stand in a line likewise. Now as technical you are, can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

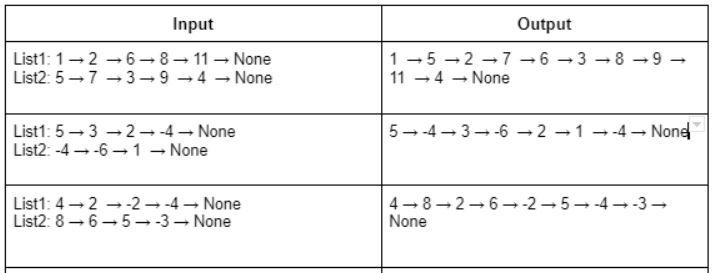
| Sample Input  10→15→34→41→56→72  Sample Input  10→15→44→41→56→72 | Sample Output  True  Sample Output  False |
| --- | --- |

1. **Word Decoder**

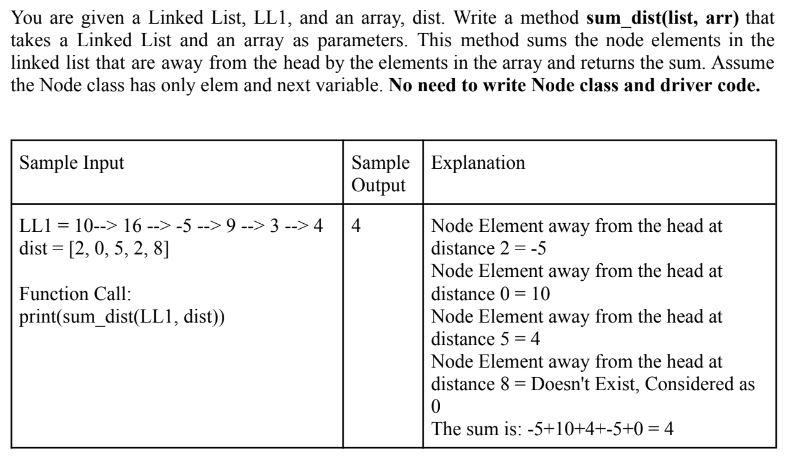
****

1. **Alternate Merge**

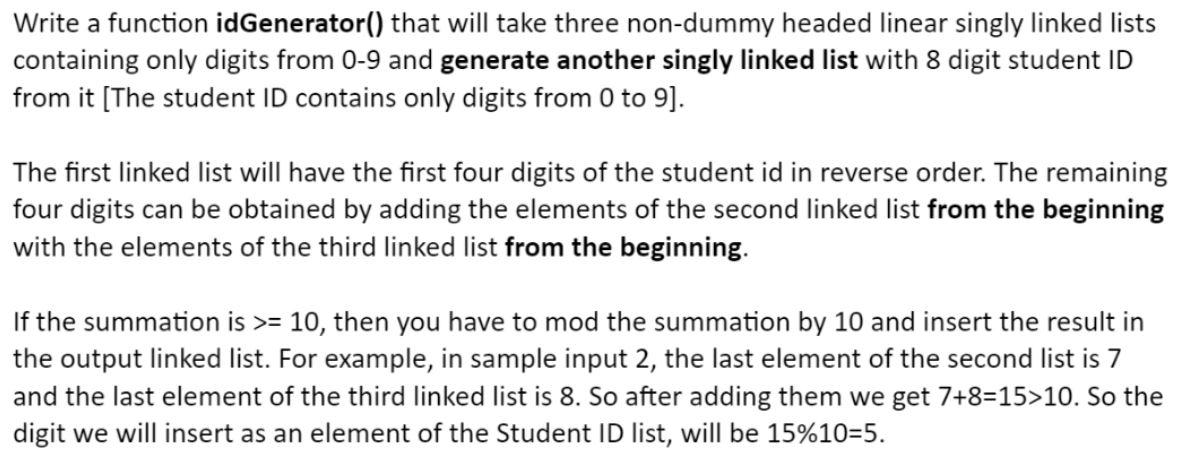
****

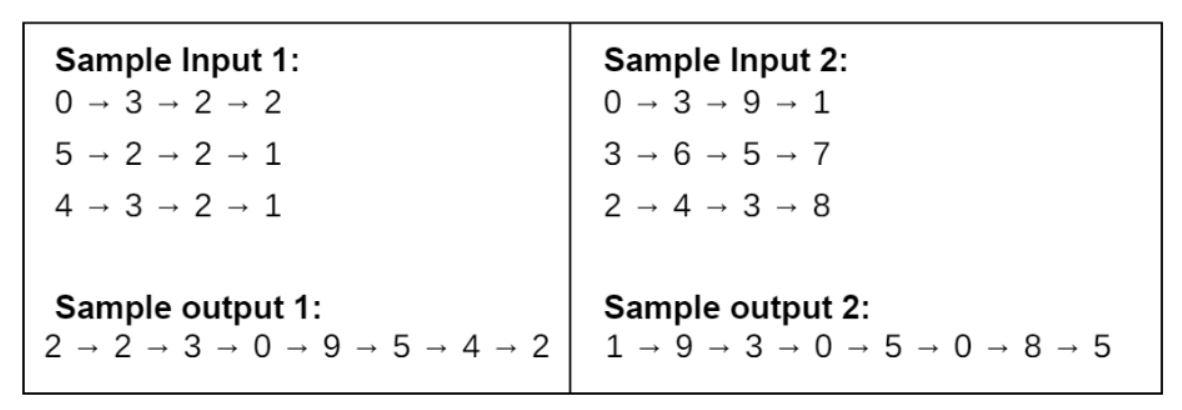
****

1. **Sum of Nodes**

****

**Bonus Task: ID Generator**

****

****