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 BUILT-IN FUNCTION EXCEPT** len **IN PYTHON. [negative indexing, append is prohibited]**
- **YOU HAVE TO MENTION SIZE OF ARRAY WHILE INITIALIZATION**
- **YOUR CODE SHOULD WORK FOR ANY VALID INPUTS. [Make changes to the Sample Inputs and check whether your program works correctly]**

Tasks on Linked List

1. 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

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.

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. Shuffle on index

In a Music player , the next button gives you the next song and the previous button moves back to the previous one. For simplicity, suppose our designed music player only has a next button. However, our music player has a special button named ‘shuffle on index’. The button creates another playlist from the given playlist where all the songs in the even index will be linked in the beginning and the odd indexed songs are linked in the end. The relative position of the song remains unchanged. Implement the ‘shuffle on index’ button that takes a playlist and returns a newly created playlist. Given playlist will contain at least two songs.

Sample Input: S→E→N→P→A→I Sample Input: N→I→S→H→I→N→O→Y→A	Sample Output: S→N→A→E→P→I Sample Output: N→S→I→O→A→I→H→N→Y

4. Scavenger hunt

You are aware of scavenger hunt right? It's basically a treasure finding game where you find a clue and each clue leads to the next clue. Suppose, you are playing a scavenger hunt where you are given a series of numbers as clues. For every clue your task is to find the first clue next to it whose value is strictly greater than it.

Sample Input: 7→85→54→16→11→30	**Sample Output:** 85 is greater than 7 No next element greater than 85 No next element greater than 54 30 is greater than 16 30 is greater than 11 No next element greater than 30
Sample Input: 20→13→33→12	**Sample Output:** 33 is greater than 20 33 is greater than 13 No next element greater than 33 No next element greater than 12