DSA LAB 2 - TUTORIAL SHEET

IMPORTANT NOTE: This is a "method-only" submission. You only need to complete the three methods. **The template files are given language wise and you have to use those files only. Do not change the name of the files and follow the format and structure given in your final submission.**
Copy the language folder of your choice in the empty repo you create for the submission and edit the file inside it.
To test on your local machine, you can make your own main function and take your own input but final submission should be in the given template format and structure.

1. PRINT ELEMENTS OF A LINKED LIST

Given a pointer to the head node of a linked list, print its elements in order, one element per line. If the head pointer is null (indicating the list is empty), don't print anything.

Input Format

The ***void PrintLinkedList(Node* head)*** method takes the head node of a linked list as a parameter. Each Node has a *'data' field* (which stores integer data) and a *'next' field* (which points to the next element in the list).
Do not read any input from stdin/console. *Each test case calls the Print method individually and passes it the head of a list.*

Output Format

Print the integer data for each element of the linked list to stdout/console.
(e.g.: using printf, cout, etc.). There should be one element per line.

Sample Input

This example uses the following two linked lists:

1. NULL

2. 1->2->3->NULL

Sample Output

1.

2. 1
   2
   3

2. INSERT A NODE AT THE TAIL OF A LINKED LIST

You are given the pointer to the head node of a linked list and an integer to add to the list. Create a new node with the given integer. Insert this node at the tail of the linked list and return the head node of the linked list formed after inserting this new node. The given head pointer may be null, meaning that the initial list is empty.

Input Format

You have to complete the ***Node* Insert(Node* head, int data)*** method. It takes two arguments: the head of the linked list and the integer to insert.
Do not read any input from the stdin/console.

Output Format

Insert the new node at the tail and just return the head of the updated linked list.
Do not print anything to stdout/console.

Sample Inputs

1. NULL, data = 2

2. 2--> NULL, data = 3

Sample Outputs

1. 2 -->NULL

2. 2 --> 3 --> NULL

3. DELETE A NODE AT A GIVEN INDEX

You’re given the pointer to the head node of a linked list and the position of a node to delete. Delete the node at the given position and return the head node. A position of 0 indicates head, a position of 1 indicates one node away from the head and so on. The list may become empty after you delete the node.

Input Format

You have to complete the ***Node* Delete(Node* head, int position)*** method which takes two arguments - the head of the linked list and the position of the node to delete.
You should NOT read any input from stdin/console. position will always be at least 0 and less than the number of the elements in the list.

Output Format

Delete the node at the given position and return the head of the updated linked list.
Do NOT print anything to stdout/console.

Sample Input

1. 1 --> 2 --> 3 --> NULL, position = 0

2. 1 --> NULL , position = 0

Sample Output

1. 2 --> 3 --> NULL

2. NULL

If you are done with submitting the 3 questions above to autolab successfully, you can attempt the following questions (not graded and not to be submitted. Just try them out on your local machines):

1. PRINT THE REVERSE OF A LINKED LIST

You are given the pointer to the head node of a linked list and you need to print all its elements in reverse order from tail to head, one element per line. The head pointer may be null meaning that the list is empty - in that case, do not print anything.

2. FIND THE INTERSECTION POINT OF TWO LINKED LISTS

There are two singly linked lists in a system. By some programming error the end node of one of the linked list got linked into the second list, forming a inverted Y shaped list. Write a program to get the point where the two linked list merge.

3. FIND IF 2 ELEMENTS OF DOUBLY LINKED LIST ADD UP TO ‘x’

Given a doubly linked list of integers, determine if there exists a pair of numbers in the linked list such that they add up to a specific target value ‘x’.