**Practice Session – October 30, 2025**

**Question 1: Insert and Traverse a Singly Linked List**

Write a C program to implement a singly linked list. The program should implement the following functions:

1. Insert a node at the beginning of the linked list.  
2. Insert a node at the middle of the linked list (after a specific position).  
3. Insert a node at the end of the linked list.  
4. Traverse the linked list and print all the elements.

Function Definitions:

1. void insertAtBeginning(struct Node\*\* head, int data);  
2. void insertAtMiddle(struct Node\*\* head, int position, int data);  
3. void insertAtEnd(struct Node\*\* head, int data);  
4. void printLinkedList(struct Node\* head);

Instructions:

- Define Node Structure with data (int) and next (pointer)  
- In main function, insert nodes at the beginning, a user-specified position and at the end.  
- Traverse the list and print all node values.

**Example 1: Insert at the Beginning**

**-** Insert 42: 42 -> NULL  
- Insert 99: 99 -> 42 -> NULL

**Example 2: Insert at the Middle**

**-** Initial List: 7 -> 13 -> NULL  
- Insert 21 at position 1: 7 -> 21 -> 13 -> NULL

**Example 3: Insert at the End**

**-** Initial List: 1 -> 3 -> NULL  
- Insert 21 at position 1: 1 -> 3 -> 77 -> NULL

**Question 2: Deleting a Node from a Singly Linked List**

Given a singly linked list and a position, implement a deletion operation. You need to determine whether the node to delete is at the **beginning**, **middle**, or **end** based on the input.

**Given Input 1:**

Input List: 5 -> 10 -> 15 -> 20 -> NULL  
Position: 4

**Expected Output:**

Modified List: 3 -> 6 -> 12 -> NULL

**Given Input 2:**

Input List: 2 -> 13 -> 7 -> 1 -> 5 -> NULL  
Position: 2

**Expected Output:**

Modified List: 2 -> 13 -> 1 -> 5 -> NULL

Instructions:

1. **Identify the Deletion Type**: Determine whether the node is deleted at the beginning, middle, or end based on given input.  
2. **Implement the Deletion**: Write the function to delete the node at the specified position.  
3. **Draw the Linked List**:

* Before Deletion: Draw the original linked list.
* After Deletion: Draw the linked list after the node is removed.

4. **Explain** the Process:

* Explain step by step how the node is removed.

**Question 3: Remove Duplicate Elements from a Sorted Linked List**

Given a sorted singly linked list, implement a function to remove all duplicate elements so that each element appears only once in the list.

**Given Input 1:**

Input LinkedList: 6 -> 6 -> 4 -> 1  
Output: 4 1

**Given Input 2:**

Input LinkedList: 2 -> 2 -> 2 -> 2 -> 2  
Output: 2

Instructions:

1. **Identify Duplicate Elements**: Traverse the list, and whenever you find two consecutive nodes with the same value, remove the second node.  
2. **Modify the Links**: Ensure the previous node points to the node after the duplicate.  
3. **Print the List**: After removing duplicates, print the updated linked list.

**Question 4: Reverse a Singly Linked List**

Given a singly linked list, implement a function to reverse the linked list so that the first element becomes the last, and the last element becomes the first.

**Given Input 1:**

Input LinkedList: 1 -> 2 -> 3 -> 4 -> 5 -> NULL

**Expected Output:**

Reversed LinkedList: 5 -> 4 -> 3 -> 2 -> 1 -> NULL

Instructions:

1. **Reverse the List**: Implement the function to reverse the pointers in the list.  
2. **Print the List**: After reversing, print the list in the reversed order.