**Delete of singly link list**

A singly linked list consists of nodes where each node contains a data field and a pointer to the next node. The operations to delete nodes include:

1️-: Delete First Node

How it works:

Check if the list is empty. If it is, return an error message.

Store the current head node in a temporary variable.

Move the head pointer to the next node.

Delete the old head node to free memory.

Why?

This ensures that the list remains connected after the first node is removed.

Time Complexity: O(1) (constant time).

2 -:Delete Last Node

How it works:

If the list is empty, return an error.

If there is only one node, delete it and set head to nullptr.

Otherwise, traverse the list to find the second-last node.

Remove the last node by setting the next pointer of the second-last node to nullptr.

Delete the last node.

Why?

This prevents access to an invalid memory location.

Time Complexity: O(n) (traverses the list to find the last node).

3️-: Delete Nth Node

How it works

If N == 1, call the delete first node function.

Otherwise, traverse the list to find the Nth node.

Keep track of the previous node to update its next pointer.

Remove the Nth node and free memory.

Why?

It allows deletion of any node at a specific position.

Time Complexity: O(n) (worst case: deleting the last node).

4️-: Delete Center Node

How it works:

Use the slow and fast pointer approach:

slow moves one step at a time.

fast moves two steps at a time.

When fast reaches the end, slow is at the middle node.

Remove the middle node by updating the previous node’s pointer.

Free the memory of the deleted node.

Why?

This method is efficient and does not require counting nodes beforehand.

Time Complexity: O(n) (traverses half the list on average).

Key Takeaways -:

Deleting the first node is the fastest operation (O(1)).

Deleting the last node requires traversal (O(n)).

Deleting the Nth node can be O(1) or O(n), depending on the position.

Deleting the center node uses the slow & fast pointer technique for efficiency.

This approach ensures efficient and safe deletion operations in a singly linked list.

