**1. deleteFirst() – Deletes the first node**

**Logic:**

- If the list is empty (head == NULL), there's nothing to delete.
- Store the current head node in a temporary pointer (temp).
- Move the head pointer to the next node (head = head->next).
- Delete the old head node (delete temp) to free up memory.

**2. deleteLast() – Deletes the last node**

**Logic:**

- If the list is empty (head == NULL), do nothing.
- If there is only one node (head->next == NULL):
    - Delete the node.
    - Set head = NULL.
- Otherwise:
    - Traverse the list to find the **second-last node** (where node->next->next == NULL).
    - Delete the last node (delete node->next).
    - Set node->next = NULL.

**3. deleteNth(int position) – Deletes the node at position N (0-based index)**

**Logic:**

- If position == 0, simply call deleteFirst().
- Otherwise:
    - Traverse the list to reach the node **just before the Nth node** (i.e., at position - 1).
    - Store the Nth node in a temporary pointer.
    - Update the (position - 1) node's next to skip the Nth node (prev->next = temp->next).
    - Delete the Nth node (delete temp).

**4. deleteCenter() – Deletes the middle node**

**Logic:**

- If the list is empty or has one node, call deleteFirst().
- Use two pointers:
    - slow moves one step at a time.
    - fast moves two steps at a time.
- Maintain a prev pointer that tracks the node **before slow**.
- When fast reaches the end, slow will be at the middle node.
- Set prev->next = slow->next to remove the middle node.
- Delete the slow node.