1. deleteFirst() – Deletes the first node

- If the list is empty (head == NULL), do nothing.
- Store the current head in a temporary pointer.
- Move head to the next node.
- Delete the old head to free memory.

2. deleteLast() – Deletes the last node

- If the list is empty, do nothing.
- If there is only one node, delete it and set head = NULL.
- Otherwise, traverse the list to find the second-last node.
- Delete the last node and update the second-last node's next pointer to NULL.

3. deleteNth(int position) – Deletes the node at position N

- If position is 0, call deleteFirst().
- Traverse to the node **before** the Nth node.
- Update its next pointer to skip the Nth node.
- Delete the Nth node to free memory.

4. deleteCenter() – Deletes the middle node

- If there is only one node, call deleteFirst().
- Use **slow and fast pointers**:
  - 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.
- Keep track of slow's previous node and update its next pointer to skip the middle node.
- Delete the middle node.

Output:

```
After deleting first node: 3->2->1->4->NULL
After deleting last node: 3->2->1->NULL
After deleting 2nd node: 3->2->NULL
After deleting center node: 3->NULL
```

Code:

```
// Delete first node
    void deleteFirst() {
       if (!head) return;
       Node* temp = head;
       head = head->next;
       delete temp;
    }
```

```
// Delete last node
void deleteLast() {
    if (!head) return;
    if (!head->next) {
        delete head;
        head = nullptr;
        return;
    }
    Node* temp = head;
    while (temp->next->next) {
        temp = temp->next;
    }
    delete temp->next;
    temp->next = nullptr;
}

// Delete Nth node
void deleteNth(int position) {
    if (!head || position < 0) return;
    if (position == 0) {
        deleteFirst();
        return;
    }
    Node* temp = head;
    for (int i = 0; temp && i < position - 1; i++) {
        temp = temp->next;
    }
    if (!temp || !temp->next) return;
    Node* nodeToDelete = temp->next;
    temp->next = temp->next->next;
    delete nodeToDelete;
}

// Delete center node
void deleteCenter() {
    if (!head || !head->next) {
        deleteFirst();
```

```
        return;
    }
    Node* slow = head;
    Node* fast = head;
    Node* prev = nullptr;
    while (fast && fast->next) {
        prev = slow;
        slow = slow->next;
        fast = fast->next->next;
    }
    if (prev) {
        prev->next = slow->next;
    }
    delete slow;
}
```