1. **displayFirst()**

- Checks if the list has at least one node (head is not NULL).
- Prints the value of the first node (head->data).

2. **displayLast()**

- If the list is empty (head == NULL), it exits immediately.
- Otherwise, it starts from the head and moves to the last node using a loop (temp = temp->next until temp->next is NULL).
- Prints the value of the last node.

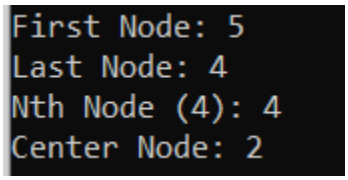
3. **displayNth(int n)**

- Starts from the head and moves forward n times.
- Uses a counter (count) to track the position.
- If the counter reaches n, it prints the value of that node.
- If n is out of bounds (list is too short), it prints "Invalid position!".

4. **displayCenter()**

- Uses two pointers:
    - slow moves one step at a time.
    - fast moves two steps at a time.
- When fast reaches the end (NULL), slow is at the middle of the list.
- Prints the value of the center node.

Output:



Code:

```
// Display the first node
    void displayFirst()
                {
      if (head)
```

```
        cout << "First Node: " << head->data << endl;
}
// Display the last node
void displayLast()
                {
   if (!head) return;
   Node* temp = head;
   while(temp->next)
                        {
      temp = temp->next;
   }
   cout << "Last Node: " << temp->data << endl;
}
// Display the Nth node (0-based index)
void displayNth(int n)
                {
   Node* temp = head;
   int count = 0;
   while(temp)
                        {
      if(count == n)
                                {
         cout << "Nth Node (" << n << "): " << temp->data << endl;
         return;
      }
      temp = temp->next;
      count++;
```

```
    }
    cout << "Invalid position!" << endl;
}

// Display the center node
void displayCenter()
                {
    if(!head) return;
    Node* slow = head;
    Node* fast = head;
    while(fast && fast->next)
                            {
        slow = slow->next;
        fast = fast->next->next;
    }
    cout << "Center Node: " << slow->data << endl;
}
```