Day 123/180 LinkedList Problems

1:Reverse a sublist of a linked list:

The function `reverseSublist()` reverses a specified sublist within a linked list. It does so by iterating through the list to identify the start and end points of the sublist, then reversing the sublist while appropriately updating pointers. Finally, it returns the head of the modified linked list.

```
  Node* reverseBetween(Node* head, int start, int end)
{
    // Handle edge cases: empty list or sublist length 1
    if (head == nullptr || start == end)
        return head;

    // Create a dummy node to handle the case when start is 1
    Node* dummy = new Node(0);
    dummy->next = head;
    Node* beforeStart = dummy;

    // Move to the (start-1)th node
    for (int i = 1; i < start; ++i) {
        beforeStart = beforeStart->next;
    }

    Node* current = beforeStart->next;
    Node* afterEnd = nullptr;

    // Reverse the linked list from start to end
    for (int i = start; i <= end; ++i) {
        Node* temp = current->next;
        current->next = afterEnd;
        afterEnd = current;
        current = temp;
    }

    // Connect the node before the start to the new head of the
```

```
reversed sublist
    beforeStart->next->next = current;
    beforeStart->next = afterEnd;

    // Return the head of the modified linked list
    return dummy->next;
}
```

2:Multiply two linked lists:

Approach 1 :
The provided code defines two functions: `reverse()` and `multiplyTwoLists()`.

The `reverse()` function reverses a linked list recursively.

The `multiplyTwoLists()` function multiplies two linked lists represented as numbers and returns the result modulo \(10^9 + 7 \). It reverses both lists, iterates through one list while multiplying it with each digit of the other list, and accumulates the result accordingly. Finally, it returns the computed result.

```
Node* reverse(Node* a)
{
    // If the current node is the last node, return it
    if(a->next == NULL)
        return a;

    // Store the next node in a temporary variable
    Node *temp = a->next;

    // Recursively reverse the rest of the linked list
    Node *head = reverse(a->next);

    // Reverse the links
    temp->next = a;
    a->next = NULL;
```

```
    // Return the new head of the reversed linked list
    return head;
}

long long multiplyTwoLists(Node* l1, Node* l2)
{
    // Reverse both input linked lists
    l1 = reverse(l1);
    l2 = reverse(l2);

    // Modulo value for calculations
    long long m = 1e9 + 7;

    // Initialize answer
    long long ans = 0;
    // Initialize multiplier
    long long mul = 1;

    // Loop through l1
    while(l1 !=  NULL)
    {
        // Get the value of the current node in l1
        long long val = l1->data;
        // Start from the beginning of l2
        Node *temp = l2;

        // Initialize multiplier for current digit
        long long n = mul % m;

        // Loop through l2
        while(temp !=  NULL)
        {
            // Update the answer using multiplication
            ans = (ans  + ((temp->data * val) * n) % m) % m;
            // Update the multiplier for the next digit
            n = (n * 10) % m;
            // Move to the next node in l2
            temp = temp->next;
```

```
        }

        // Update the multiplier for the next digit
        mul = (mul * 10) % m;
        // Move to the next node in l1
        l1 = l1->next;
    }

    // Return the final answer
    return ans;
}
```

Approach 2 :
Convert the LinkedList to a number, then multiply and return the answer.

```
#define mod 1000000007

long long convert(Node *head) {
    long long num = 0;
    Node *curr = head;

    while (curr) {
        num = (num * 10 + curr->data) % mod;
        curr = curr->next;
    }

    return num;
}

long long  multiplyTwoLists (Node* l1, Node* l2)
{
    long long num1 = convert(l1)%mod;
    long long num2 = convert(l2)%mod;

    return (num1*num2)%mod;
}
```