Merge Sort on Linked List LeetCode

You are given an unsorted linked list. Write a C++ program to sort the linked list using the Merge Sort algorithm. Implement the Merge Sort approach

Example:

6 -> 4 -> 2 -> 3 -> 3 -> 2 -> 1

Output:

1 -> 2 -> 2 -> 3 -> 3 -> 4 -> 6

Function to find the middle using the two-pointer approach (findMid)

- Finds the middle node of the linked list using the two-pointer approach.
- The slow pointer moves one step while the fast pointer moves two steps.
- When the fast pointer reaches the end, the slow pointer is at the middle node.
- **Time Complexity: O(n), where n is the number of nodes in the linked list.**
- **Space Complexity: O(1).**

Function to Merge Sort on Linked List Iterative approach (mergeSort)

- Base case: If the list is empty or has only one element, it's already sorted.
- Split the linked list into individual nodes (sub-lists) containing one node each.
- For each pair of adjacent nodes, merge them using the **mergeTwoLists** function.
- Repeat the merging process iteratively until a single sorted list is obtained.
- **Time Complexity: O(n log n), where n is the number of nodes in the linked list.**
- **Space Complexity: O(1), as the merging is done in-place without additional memory.**

Function to Merge Sort on Linked List Recursive approach (mergeSort)

- Base case: If the list is empty or has only one element, it's already sorted.
- Find the middle node of the linked list using the **findMid** function.
- Split the linked list into two halves: left and right.
- Recursively sort the left and right halves.
- Merge the sorted left and right halves using the **mergeTwoLists** function.

- **Time Complexity: O(n log n), where n is the number of nodes in the linked list.**
- **Space Complexity: O(log n) for the recursive call stack.**

Function to merge two linked lists Recursively (mergeTwoListsRecursively)

- Initialize pointers for the merged list's head and tail.
- Compare the values of the first nodes of both lists.
- Add the smaller value node to the merged list and move to the next node in that list.
- Repeat the comparison and addition process until one of the lists becomes empty.
- If any nodes are remaining in the first list, add them to the merged list.
- If any nodes are remaining in the second list, add them to the merged list.
- **Time Complexity: O(n + m), where n and m are the lengths of the two lists being merged.**
- **Space Complexity: O(1), as the merging is done in-place without additional memory.**

Function to merge two linked lists Iteratively (mergeTwoLists)

- Base cases: If one of the lists is empty, return the other list.
- Compare values of the first nodes of both lists.
- Merge the lists by appending the smaller value node to the merged list.
- Recursively merge the rest of the lists by moving to the next node of the merged list and the smaller value list.
- **Time Complexity: O(n + m), where n and m are the lengths of the two lists being merged.**
- **Space Complexity: O(n + m) due to the recursive call stack.**