Puzzle: Reverse a Linked List in Groups

You are given a linked list and a positive integer `k`. Your task is to reverse every group of `k` nodes in the linked list. If there are fewer than `k` nodes remaining at the end of the list, leave them as they are.

Details:

1. **Input**:
   - A singly linked list.
   - An integer `k`, which specifies the size of each group to be reversed.
2. **Output**:
   - The head of the modified linked list after reversing the nodes in each group of size `k`.

Example:

- Given the linked list `1 -> 2 -> 3 -> 4 -> 5` and `k = 3`, the output should be `3 -> 2 -> 1 -> 4 -> 5`. The first group of `k=3` nodes (`1 -> 2 -> 3`) is reversed, while the remaining nodes (`4 -> 5`) are left unchanged.
- Given the linked list `1 -> 2 -> 3 -> 4 -> 5` and `k = 2`, the output should be `2 -> 1 -> 4 -> 3 -> 5`. The nodes are reversed in pairs.

Example Inputs and Outputs:

1. `reverse_k_group(1 -> 2 -> 3 -> 4 -> 5, 3)` should return `3 -> 2 -> 1 -> 4 -> 5`.
2. `reverse_k_group(1 -> 2 -> 3 -> 4 -> 5, 2)` should return `2 -> 1 -> 4 -> 3 -> 5`.
3. `reverse_k_group(1 -> 2 -> 3 -> 4 -> 5, 5)` should return `5 -> 4 -> 3 -> 2 -> 1`.

Hints:

- You might need to use extra pointers to handle the reversal of nodes in groups.
- Consider how to connect the reversed groups with the rest of the list.

Puzzle: Merge Two Sorted Linked Lists

You are given two linked lists, each sorted in ascending order. Your task is to merge these two linked lists into a single sorted linked list.

Details:

1. **Input**:

    - Two singly linked lists, `list1` and `list2`, where each list is sorted in ascending order.
2. **Output**:
    - A single sorted linked list that contains all the nodes from `list1` and `list2`, also sorted in ascending order.

Example:

- Given the linked lists:
    - `list1: 1 -> 3 -> 5`
    - `list2: 2 -> 4 -> 6`

  The merged linked list should be:

    - `1 -> 2 -> 3 -> 4 -> 5 -> 6`
- Given the linked lists:
    - `list1: 1 -> 2 -> 3`
    - `list2: 4 -> 5 -> 6`

  The merged linked list should be:

    - `1 -> 2 -> 3 -> 4 -> 5 -> 6`

Example Inputs and Outputs:

1. `merge_sorted_lists(1 -> 3 -> 5, 2 -> 4 -> 6)` should return `1 -> 2 -> 3 -> 4 -> 5 -> 6`.
2. `merge_sorted_lists(1 -> 2 -> 3, 4 -> 5 -> 6)` should return `1 -> 2 -> 3 -> 4 -> 5 -> 6`.
3. `merge_sorted_lists(1 -> 5, 2 -> 3 -> 4)` should return `1 -> 2 -> 3 -> 4 -> 5`.

Hints:

- Use two pointers to traverse each linked list and compare nodes to build the merged list.
- Be sure to handle cases where one list is empty.