LeetCode Linked List: AddTwoNumbers & MergeTwoLists

Problem 1: Add Two Numbers (Linked List)

Description:

Given two numbers represented by linked lists in reverse order, add them as you would do digit by digit.

Example:

Input: l1 = [2,4,3], l2 = [5,6,4]

Output: [7,0,8] because 342 + 465 = 807

Code with Steps:

```
public class Solution {
    public ListNode AddTwoNumbers(ListNode l1, ListNode l2) {
        ListNode dummyHead = new ListNode(0);     // Step 1: create dummy head to simplify result list creation
        ListNode current = dummyHead;             // Step 2: current points to last node in result list
        int carry = 0;                            // Step 3: carry is used when sum >= 10

        while (l1 != null || l2 != null || carry != 0) {
            int sum = carry;                      // Step 4: Start with carry from last digit

            if (l1 != null) {
                sum += l1.val;                    // Step 5: add l1's digit if available
                l1 = l1.next;                     // Step 6: move to next node in l1
```

```
            }

            if (l2 != null) {
                sum += l2.val;                  // Step 7: add l2's digit if available
                l2 = l2.next;                   // Step 8: move to next node in l2
            }

            carry = sum / 10;                   // Step 9: calculate carry for next digit
            int digit = sum % 10;               // Step 10: extract current digit

            current.next = new ListNode(digit);   // Step 11: add new node with the digit
            current = current.next;             // Step 12: move current forward
        }

        return dummyHead.next;                  // Step 13: return the list starting after dummy node
    }
}
```

Problem 2: Merge Two Sorted Lists

Description:

You are given two sorted linked lists. Merge them into one sorted list.

Example:

Input: l1 = [1,2,4], l2 = [1,3,4]

Output: [1,1,2,3,4,4]

Code with Steps:

```
public class Solution {
    public ListNode MergeTwoLists(ListNode l1, ListNode l2) {
        ListNode dummyHead = new ListNode(0);     // Step 1: dummy head helps build result
        ListNode current = dummyHead;             // Step 2: pointer to build new list

        while (l1 != null && l2 != null) {
            if (l1.val < l2.val) {
                current.next = l1;                // Step 3: attach l1's node if smaller
                l1 = l1.next;                     // Step 4: move l1 forward
            } else {
                current.next = l2;                // Step 5: attach l2's node if smaller or equal
                l2 = l2.next;                     // Step 6: move l2 forward
            }
            current = current.next;               // Step 7: move current forward
        }

        if (l1 != null) current.next = l1;        // Step 8: attach remaining l1 nodes
        if (l2 != null) current.next = l2;        // Step 9: attach remaining l2 nodes

        return dummyHead.next;                    // Step 10: return result starting after dummy
    }
}
```