You are given two **non-empty** linked lists representing two non-negative integers. The digits are stored in **reverse order**, and each of their nodes contains a single digit. Add the two numbers and return the sum as a linked list.

You may assume the two numbers do not contain any leading zero, except the number 0 itself.

Example 1:

Input: l1 = [2,4,3], l2 = [5,6,4] **Output:** [7,0,8]
Explanation: 342 + 465 = 807.

Example 2:

Input: l1 = [0], l2 = [0] **Output:** [0]

Example 3:

Input: l1 = [9,9,9,9,9,9,9], l2 = [9,9,9,9] **Output:** [8,9,9,9,0,0,0,1]

Constraints:

The number of nodes in each linked list is in the range [1, 100].
0 <= Node.val <= 9
It is guaranteed that the list represents a number that does not have leading zeros.

Solution:-

```
class Solution {
    public ListNode addTwoNumbers(ListNode l1, ListNode l2)
     {
        // Head of the new linked list - this is the head of the
resultant list
        ListNode head = null;
        // Reference of head which is null at this point
        ListNode temp = null;
        // Carry
        int carry = 0;
        // Loop for the two lists
        while (l1 != null || l2 != null) {
            // At the start of each iteration, we should add carry
from the last iteration
            int sum = carry;
            // Since the lengths of the lists may be unequal, we are
checking if the
```

```
            // current node is null for one of the lists
            if (l1 != null) {
                sum += l1.val;
                l1 = l1.next;
            }
            if (l2 != null) {
                sum += l2.val;
                l2 = l2.next;
            }
            // At this point, we will add the total sum % 10 to the
new node
            // in the resultant list
            ListNode node = new ListNode(sum % 10);
            // Carry to be added in the next iteration
            carry = sum / 10;
            // If this is the first node or head
            if (temp == null) {
                temp = head = node;
            }
            // For any other node
            else {
                temp.next = node;
                temp = temp.next;
            }
        }
        // After the last iteration, we will check if there is carry
left
        // If it's left then we will create a new node and add it
        if (carry > 0) {
            temp.next = new ListNode(carry);
        }
        return head;
    }
}
```