Question :
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 ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next

def addTwoNumbers(l1, l2):
    dummy = ListNode()
    current = dummy
    carry = 0

    while l1 or l2 or carry:
        sum = carry
    if l1:
        sum += l1.val
        l1 = l1.next

    if l2:
        sum += l2.val
        l2 = l2.next

    carry = sum // 10
    digit = sum % 10

    current.next = ListNode(digit)
    current = current.next

return dummy.next
```