2. 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.

ANS -

To solve this problem, we can traverse both linked lists simultaneously, starting from the heads of the lists. We'll perform the following steps:

Create a dummy node and initialize it as the head of the result linked list.

Initialize variables carry and sum as 0 to keep track of the carry value and the sum of corresponding digits from the two lists.

Traverse both linked lists until both lists and the carry value are exhausted:

Set x as the value of the current node in the first linked list (or 0 if the first list is exhausted).

Set y as the value of the current node in the second linked list (or 0 if the second list is exhausted).

Calculate the sum of x, y, and the carry value as sum = x + y + carry.

Update the carry value as carry = sum // 10 (integer division).

Create a new node with the value of sum % 10 and append it to the result linked list.

Move to the next nodes in both lists.

After the traversal, if the carry value is still non-zero, append a new node with the value of the carry to the result linked list.

Return the next node of the dummy node as the head of the resulting linked list.

Here is the Python implementation of the above steps:

class ListNode:

def \_\_init\_\_(self, val=0, next=None):

self.val = val

self.next = next

def addTwoNumbers(l1, l2):

dummy = ListNode() # Dummy node to initialize the result linked list

curr = dummy # Pointer to traverse the result linked list

carry = 0 # Carry value

while l1 or l2 or carry:

x = l1.val if l1 else 0

y = l2.val if l2 else 0

sum = x + y + carry

carry = sum // 10

curr.next = ListNode(sum % 10)

curr = curr.next

# Move to the next nodes in both lists

l1 = l1.next if l1 else None

l2 = l2.next if l2 else None

return dummy.next # Return the head of the resulting linked list

Test cases

Create the linked lists for the given examples

l1 = ListNode(2)

l1.next = ListNode(4)

l1.next.next = ListNode(3)

l2 = ListNode(5)

l2.next = ListNode(6)

l2.next.next = ListNode(4)

result = addTwoNumbers(l1, l2)

Traverse and print the resulting linked list

while result:

print(result.val, end=" ")

result = result.next

Output: 7 0 8

The implementation adds the two given linked lists representing non-negative integers in reverse order and returns the result as a linked list. The solution handles the given examples and meets the provided constraints.