

# DAY 2

**2. Add Two Numbers**

[Visualize](#) [Topics](#) [Companies](#)

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:**

```
graph LR
    subgraph List1 [List 1]
        direction LR
        n1_1((2)) --> n1_2((4))
        n1_2 --> n1_3((3))
    end
    subgraph List2 [List 2]
        direction LR
        n2_1((5)) --> n2_2((6))
        n2_2 --> n2_3((4))
    end
    subgraph Result [Result]
        direction LR
        r1((7)) --> r2((0))
        r2 --> r3((8))
    end
```

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

**Example 2:**  
**Inputs:** l1 = [9], l2 = [9]  
**Output:** [8]

**Example 3:**  
**Inputs:** l1 = [9,9,9,9,9,9,9], l2 = [9,9,9,9]  
**Output:** [8,9,9,9,0,0,1]

## CODE:

```
class Solution {  
  
    public ListNode addTwoNumbers(ListNode l1, ListNode l2) {  
  
        ListNode dummyHead = new ListNode(0);  
  
        ListNode tail = dummyHead;  
  
        int carry = 0;  
  
        while (l1 != null || l2 != null || carry != 0) {  
  
            int digit1 = (l1 != null) ? l1.val : 0;  
  
            int digit2 = (l2 != null) ? l2.val : 0;  
  
            int sum = digit1 + digit2 + carry;  
  
            int digit = sum % 10;  
  
            carry = sum / 10;  
  
            ListNode newNode = new ListNode(digit);  
  
            tail.next = newNode;  
  
            tail = tail.next;  
  
            l1 = (l1 != null) ? l1.next : null;  
  
            l2 = (l2 != null) ? l2.next : null;  
  
        }  
  
        ListNode result = dummyHead.next;  
  
        dummyHead.next = null;  
  
        return result;  
  
    }  
}
```