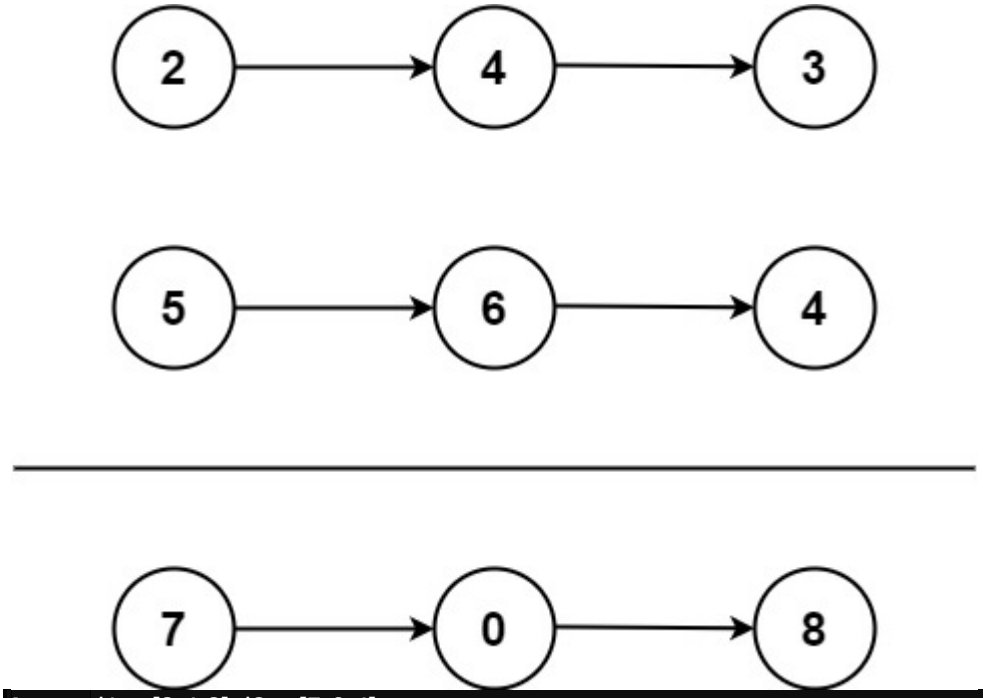
LeetCode Problem (LinkedList):

ou 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:

```
#include <stdio.h>
#include <stdlib.h>

// Definition for singly-linked list.
struct ListNode {
    int val;
    struct ListNode *next;
};

struct ListNode* createNode(int val) {
    struct ListNode* newNode = (struct ListNode*)malloc(sizeof(struct ListNode));
    newNode->val = val;
    newNode->next = NULL;
    return newNode;
}

struct ListNode* addTwoNumbers(struct ListNode* l1, struct ListNode* l2) {
    // Initialize a dummy node to keep track of the head of the result list
    struct ListNode dummyHead;
    dummyHead.next = NULL;
    struct ListNode* current = &dummyHead;
    int carry = 0;

    // Loop through lists l1 and l2 until you reach both ends, along with carry.
    while (l1 != NULL || l2 != NULL || carry != 0) {
        // Get the values from the current nodes, if available
        int l1_val = (l1 != NULL) ? l1->val : 0;
        int l2_val = (l2 != NULL) ? l2->val : 0;
```

```
        // Calculate the sum and update the carry
        int total = l1_val + l2_val + carry;
        carry = total / 10;
        int new_val = total % 10;

        // Create a new node with the sum's digit and move the pointer
        current->next = createNode(new_val);
        current = current->next;

        // Move to the next nodes in the input lists, if available
        if (l1 != NULL) l1 = l1->next;
        if (l2 != NULL) l2 = l2->next;
    }

    // Return the head of the new linked list
    return dummyHead.next;
}

// Function to print the linked list
void printLinkedList(struct ListNode* node) {
    while (node != NULL) {
        printf("%d", node->val);
        if (node->next != NULL) printf(" -> ");
        node = node->next;
    }
    printf("\n");
}

// Helper function to create a linked list from an array
struct ListNode* createLinkedList(int* arr, int size) {
```

```
    struct ListNode* head = NULL;
    struct ListNode* current = NULL;
    for (int i = 0; i < size; i++) {
        struct ListNode* newNode = createNode(arr[i]);
        if (head == NULL) {
            head = newNode;
            current = newNode;
        } else {
            current->next = newNode;
            current = newNode;
        }
    }
    return head;
}

int main() {
    // Example usage:
    int arr1[] = {2, 4, 3};
    int arr2[] = {5, 6, 4};
    struct ListNode* l1 = createLinkedList(arr1, 3);
    struct ListNode* l2 = createLinkedList(arr2, 3);

    struct ListNode* result = addTwoNumbers(l1, l2);
    printLinkedList(result);

    // Free allocated memory (not shown for simplicity)

    return 0;
}
```