2.Add Two Numbers

> 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 numbeer 0 itself.

My Anwser

Difficulty	Status	Runtime	Distribution(%)	Memory	Distribution(%)
Medium	AC	18ms	55.99	7.9MB	29.19

```
struct ListNode* addTwoNumbers(struct ListNode* l1, struct ListNode* l2){
    struct ListNode* head = NULL;
    struct ListNode* tail = NULL;
    int num1, num2, sum;
    int carry = 0;
    while (l1 || l2) {
        num1 = (l1) ? l1->val : 0;
        num2 = (l2) ? l2->val : 0;
        sum = num1 + num2 + carry;
        carry = sum / 10;
        if (!head) {
            head = (struct ListNode*)malloc(sizeof(struct ListNode));
            tail = head;
            head->val = sum % 10;
            head->next = NULL;
        } else {
            tail->next = (struct ListNode*)malloc(sizeof(struct ListNode));
            tail = tail->next;
            tail->val = sum % 10;
            tail->next = NULL;
        }
        if (l1) {
            l1 = l1->next;
        }
        if (l2) {
            l2 = l2->next;
        }
    }
    if (carry > 0) {
        tail->next = (struct ListNode*)malloc(sizeof(struct ListNode));
        tail = tail->next;
        tail->val = carry;
        tail->next = NULL;
    }
    return head;
}
```

Algorithm

Time complexity	Space complexity
$O(n)$	$O(n)$

Keep track of the carry using a variable and simulate digits-by-digits sum starting from the head of the list,which contains the least-significant digit.