1) **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 contain a single digit. Add the two numbers and return it as a linked list.

You may assume the two numbers do not contain any leading zero, except the number 0 itself.

Example:

```
Input: (2 -> 4 -> 3) + (5 -> 6 -> 4)
Output: 7 -> 0 -> 8
Explanation: 342 + 465 = 807.
```

Solution:

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

Figure 1. Visualization of the addition of two numbers: $342 + 465 = 807$342+465=807. Each node contains a single digit and the digits are stored in reverse order.

Algorithm

Just like how you would sum two numbers on a piece of paper, we begin by summing the least-significant digits, which is the head of $l1$l1 and $l2$l2. Since each digit is in the range of $0 \ldots 9$0…9, summing two digits may "overflow". For example $5 + 7 = 12$5+7=12. In this case, we set the current digit to $2$2 and bring over the $\text{carry} = 1$carry=1 to the next iteration. carrycarry must be either $0$0 or $1$1 because the largest possible sum of two digits (including the carry) is $9 + 9 + 1 = 19$9+9+1=19.

The pseudocode is as following:

- Initialize current node to dummy head of the returning list.
- Initialize carry to $0$0.
- Initialize pp and qq to head of $l1$l1 and $l2$l2 respectively.
- Loop through lists $l1$l1 and $l2$l2 until you reach both ends.
    - Set xx to node pp's value. If pp has reached the end of $l1$l1, set to $0$0.
    - Set yy to node qq's value. If qq has reached the end of $l2$l2, set to $0$0.
    - Set $\text{sum} = x + y + \text{carry}$sum=x+y+carry.
    - Update $\text{carry} = \text{sum} / 10$carry=sum/10.
    - Create a new node with the digit value of $(\text{sum} \bmod 10)$(summod10) and set it to current node's next, then advance current node to next.
    - Advance both pp and qq.
- Check if $\text{carry} = 1$carry=1, if so append a new node with digit $1$1 to the returning list.
- Return dummy head's next node.

Code:

C Code

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


struct ListNode* addTwoNumbers(struct ListNode* l1, struct ListNode* l2){
    int carry=0;
    int sum=0;
    struct ListNode* ret=NULL;
    struct ListNode* tmp=NULL;
    struct ListNode* curr;
    while(l1 || l2){
        int num1=0;
        int num2=0;
        if(l1){
            num1=l1->val;
            l1=l1->next;
        }
        if(l2){
            num2=l2->val;
            l2=l2->next;
        }
        sum=num1+num2+carry;

        carry=sum/10;
        sum=sum%10;

        tmp=(struct ListNode*)malloc(sizeof(struct ListNode));

        if(NULL == ret){
            ret=tmp;
        }
        tmp->val=sum;
        tmp->next=NULL;
        curr->next=tmp;
        curr=tmp;


    }
```

```
    if(carry){
        tmp=(struct ListNode*)malloc(sizeof(struct ListNode));
        tmp->val=carry;
        tmp->next=NULL;
        curr->next=tmp;

    }

    return ret;

}
```

C++ code

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    ListNode* addTwoNumbers(ListNode* l1, ListNode* l2) {
        int carry=0;
        int sum=0;
        ListNode* ret=NULL;
        ListNode* tmp=NULL;
        ListNode* curr;
        while(l1 || l2){
            int num1=0;
            int num2=0;
            if(l1){
                num1=l1->val;
                l1=l1->next;
            }
            if(l2){
                num2=l2->val;
                l2=l2->next;
            }
            sum=num1+num2+carry;

            carry=sum/10;
```

```
            sum=sum%10;

            tmp=new ListNode(sum);

            if(NULL == ret){
               ret=tmp;
            }
            if(curr){
               curr->next=tmp;
            }
            curr=tmp;


         }

         if(carry){
            tmp=new ListNode(carry);
            curr->next=tmp;

         }

         return ret;
      }
};
```