2. Add Two Numbers

- 注意 list 是否為 NULL
- 兩個 list 開始相加，紀錄進位
- 要更省記憶體的話就是要沿用原本 `l1 or l2` 的空間，只有最後的進位才新增新空間

```c
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     struct ListNode *next;
 * };
 */
#include <stdio.h>
#include <stdlib.h>

struct ListNode* addTwoNumbers(struct ListNode* l1, struct ListNode* l2)
{

    unsigned int sum = 0, c = 0;
    struct ListNode *head = NULL;
    struct ListNode *cur = NULL;


    while ((l1 != NULL) || (l2 != NULL) || c) {
        struct ListNode *new = calloc(sizeof(struct ListNode), 1);

        sum = c;

        if (l1 != NULL) {
            sum += l1->val;
            l1 = l1->next;
        }

        if (l2 != NULL) {
            sum += l2->val;
            l2 = l2->next;
        }

        (sum >= 10) ? (sum -= 10, c = 1) : (c = 0);
        new->val = sum;
        new->next = NULL;

        if (cur == NULL) {
            head = new;
            cur = head;
        } else {
            cur->next = new;
            cur = cur->next;
        }
    }

    return head;
}
```

- 省記憶體的方式

```c
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     struct ListNode *next;
 * };
 */
#include <stdio.h>
#include <stdlib.h>

static struct ListNode* addTwoNumbers(struct ListNode* l1, struct ListNode* l2)
{
    int carry = 0;
    struct ListNode dummy;
    struct ListNode *p = l1, *prev = &dummy;

    dummy.next = p;
    while (l1 != NULL || l2 != NULL) {
        int sum = 0;

        if (l1 != NULL) {
            sum += l1->val;
            l1 = l1->next;
        }

        if (l2 != NULL) {
            if (p == NULL) {
                /* l2 longer than l1 */
                prev->next = l2;
                p = l2;
            }
            sum += l2->val;
            l2 = l2->next;
        }

        sum += carry;
        carry = sum / 10;
        p->val = sum % 10;
        prev = p;
        p = p->next;
    }

    if (carry) {
        p = malloc(sizeof(*p));
        p->val = carry;
        p->next = NULL;
        prev->next = p;
    }

    return dummy.next;
}
```