

## 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

```
1  /**
2   * Definition for singly-linked list.
3   * struct ListNode {
4   *     int val;
5   *     struct ListNode *next;
6   * };
7   */
8  #include <stdio.h>
9  #include <stdlib.h>
10
11 struct ListNode* addTwoNumbers(struct ListNode* l1, struct ListNode* l2)
12 {
13
14     unsigned int sum = 0, c = 0;
15     struct ListNode *head = NULL;
16     struct ListNode *cur = NULL;
17
18
19     while ((l1 != NULL) || (l2 != NULL) || c) {
20         struct ListNode *new = calloc(sizeof(struct ListNode), 1);
21
22         sum = c;
23
24         if (l1 != NULL) {
25             sum += l1->val;
26             l1 = l1->next;
27         }
28
29         if (l2 != NULL) {
30             sum += l2->val;
31             l2 = l2->next;
32         }
33
34         (sum >= 10) ? (sum -= 10, c = 1) : (c = 0);
35         new->val = sum;
36         new->next = NULL;
37
38         if (cur == NULL) {
39             head = new;
40             cur = head;
41         } else {
42             cur->next = new;
43             cur = cur->next;
44         }
45     }
46
47     return head;
48 }
```

- 省記憶體的方式

```
1  /**
2   * Definition for singly-linked list.
3   * struct ListNode {
4   *     int val;
5   *     struct ListNode *next;
6   * };
7   */
8  #include <stdio.h>
9  #include <stdlib.h>
10
11 static struct ListNode* addTwoNumbers(struct ListNode* l1, struct ListNode*
12 l2)
13 {
14     int carry = 0;
15     struct ListNode dummy;
16     struct ListNode *p = l1, *prev = &dummy;
17
18     dummy.next = p;
19     while (l1 != NULL || l2 != NULL) {
20         int sum = 0;
21
22         if (l1 != NULL) {
23             sum += l1->val;
24             l1 = l1->next;
25         }
26
27         if (l2 != NULL) {
28             if (p == NULL) {
29                 /* l2 longer than l1 */
30                 prev->next = l2;
31                 p = l2;
32             }
33             sum += l2->val;
34             l2 = l2->next;
35
36             sum += carry;
37             carry = sum / 10;
38             p->val = sum % 10;
39             prev = p;
40             p = p->next;
41         }
42
43         if (carry) {
44             p = malloc(sizeof(*p));
45             p->val = carry;
46             p->next = NULL;
47             prev->next = p;
48         }
49
50         return dummy.next;
51     }
```