原题：

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.

**Input:** (2 -> 4 -> 3) + (5 -> 6 -> 4)  
**Output:** 7 -> 0 -> 8

分析：

依次遍历两个链表，新建节点保存两个数值的模10结果，除10结果用于进位。若某一链表有剩余，继续用同样方法处理剩余部分。

/\*\*

\* Definition for singly-linked list.

\* struct ListNode {

\* int val;

\* ListNode \*next;

\* ListNode(int x) : val(x), next(NULL) {}

\* };

\*/

class Solution {

public:

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

ListNode \*dummy = new ListNode(-1);

ListNode \*pre = dummy;

ListNode \*p1 = l1;

ListNode \*p2 = l2;

int m = 0;

int n = 0;

while (p1 != NULL && p2 != NULL) {

m = (p1->val + p2->val + n) % 10;

ListNode \*node = new ListNode(-1);

node->val = m;

pre->next = node;

pre = node;

n = (p1->val + p2->val + n) / 10;

p1 = p1->next;

p2 = p2->next;

}

while (p1 != NULL) {

m = (p1->val + n) % 10;

ListNode \*node = new ListNode(-1);

node->val = m;

pre->next = node;

pre = node;

n = (p1->val + n) / 10;

p1 = p1->next;

}

while (p2 != NULL) {

m = (p2->val + n) % 10;

ListNode \*node = new ListNode(-1);

node->val = m;

pre->next = node;

pre = node;

n = (p2->val + n) / 10;

p2 = p2->next;

}

if (n > 0) {

ListNode \*node = new ListNode(-1);

node->val = n;

pre->next = node;

}

return dummy->next;

}

};