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.
```

Original Quiz in C++

```
/**
 * 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) {

    }
};
```

Solution 1 - C++

twonumbers.cpp

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode(int x) : val(x), next(NULL) {}
 * };
 */

#define MNEXT(A) (A?(A->next?A->next:NULL):NULL)
#define MSELECT(A,B) (A?A:(B?B:NULL))

class Solution {
public:
    ListNode* addTwoNumbers(ListNode* l1, ListNode* l2) {
        if (!l1 && !l2) return NULL;

        ListNode *n=new ListNode(0);
        if (l1) n->val+=l1->val;
        if (l2) n->val+=l2->val;
        if (n->val>9)
        {
            ListNode *next;
            next=MNEXT(l1);
            if (!next) next=MNEXT(l2);

            if (next) // next node(s) exists, so carry over 1 to next digit
            {
              next->val += 1;
            }
            else // no next node found, so create new node with initial value=1
            {
                ListNode *p=MSELECT(l1,l2);
                p->next=new ListNode(1);
            }
            n->val -=10;
        }
        if (MNEXT(l1) || MNEXT(l2))
            n->next = addTwoNumbers(MNEXT(l1), MNEXT(l2));
        return n;
    }
};
```

Solution 2 in Java

add_two_numbers.java

```
public ListNode addTwoNumbers(ListNode l1, ListNode l2) {
    ListNode dummyHead = new ListNode(0);
    ListNode p = l1, q = l2, curr = dummyHead;
    int carry = 0;
    while (p != null || q != null) {
        int x = (p != null) ? p.val : 0;
        int y = (q != null) ? q.val : 0;
        int sum = carry + x + y;
        carry = sum / 10;
        curr.next = new ListNode(sum % 10);
        curr = curr.next;
        if (p != null) p = p.next;
        if (q != null) q = q.next;
    }
    if (carry > 0) {
        curr.next = new ListNode(carry);
    }
    return dummyHead.next;
}
```