**Question**

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.

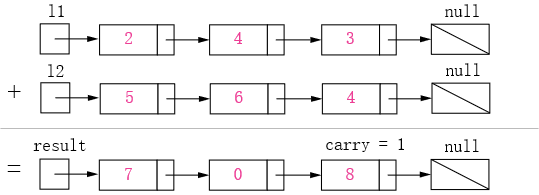
You may assume the two numbers do not contain any leading zero, except the number 0 itself.

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

**Solution**

**Intuition**

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=807342 + 465 = 807342+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 l1l1l1 and l2l2l2. Since each digit is in the range of 0…90 \ldots 90…9, summing two digits may "overflow". For example 5+7=125 + 7 = 125+7=12. In this case, we set the current digit to 222 and bring over the carry=1carry = 1carry=1 to the next iteration. carrycarrycarry must be either 000 or 111 because the largest possible sum of two digits (including the carry) is 9+9+1=199 + 9 + 1 = 199+9+1=19.

The pseudocode is as following:

* Initialize current node to dummy head of the returning list.
* Initialize carry to 000.
* Initialize ppp and qqq to head of l1l1l1 and l2l2l2 respectively.
* Loop through lists l1l1l1 and l2l2l2 until you reach both ends.
  + Set xxx to node ppp's value. If ppp has reached the end of l1l1l1, set to 000.
  + Set yyy to node qqq's value. If qqq has reached the end of l2l2l2, set to 000.
  + Set sum=x+y+carrysum = x + y + carrysum=x+y+carry.
  + Update carry=sum/10carry = sum / 10carry=sum/10.
  + Create a new node with the digit value of (summod10)(summod10) and set it to current node's next, then advance current node to next.
  + Advance both ppp and qqq.
* Check if carry=1carry = 1carry=1, if so append a new node with digit 111 to the returning list.
* Return dummy head's next node.

Note that we use a dummy head to simplify the code. Without a dummy head, you would have to write extra conditional statements to initialize the head's value.

Take extra caution of the following cases:

| **Test case** | **Explanation** |
| --- | --- |
| l1=[0,1]l1=[0,1]l1=[0,1] l2=[0,1,2]l2=[0,1,2]l2=[0,1,2] | When one list is longer than the other. |
| l1=[]l1=[]l1=[] l2=[0,1]l2=[0,1]l2=[0,1] | When one list is null, which means an empty list. |
| l1=[9,9]l1=[9,9]l1=[9,9] l2=[1]l2=[1]l2=[1] | The sum could have an extra carry of one at the end, which is easy to forget. |

**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**;**

**}**

**Complexity Analysis**

* Time complexity : O(max(m,n))O(\max(m, n))O(max(m,n)). Assume that mmm and nnn represents the length of l1l1l1 and l2l2l2 respectively, the algorithm above iterates at most max(m,n)\max(m, n)max(m,n) times.
* Space complexity : O(max(m,n))O(\max(m, n))O(max(m,n)). The length of the new list is at most max(m,n)+1\max(m,n) + 1max(m,n)+1.

**Follow up**

What if the the digits in the linked list are stored in non-reversed order? For example:

(3→4→2)+(4→6→5)=8→0→7 (3 \to 4 \to 2) + (4 \to 6 \to 5) = 8 \to 0 \to 7 (3→4→2)+(4→6→5)=8→0→7