Add Two Numbers

Problem

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.

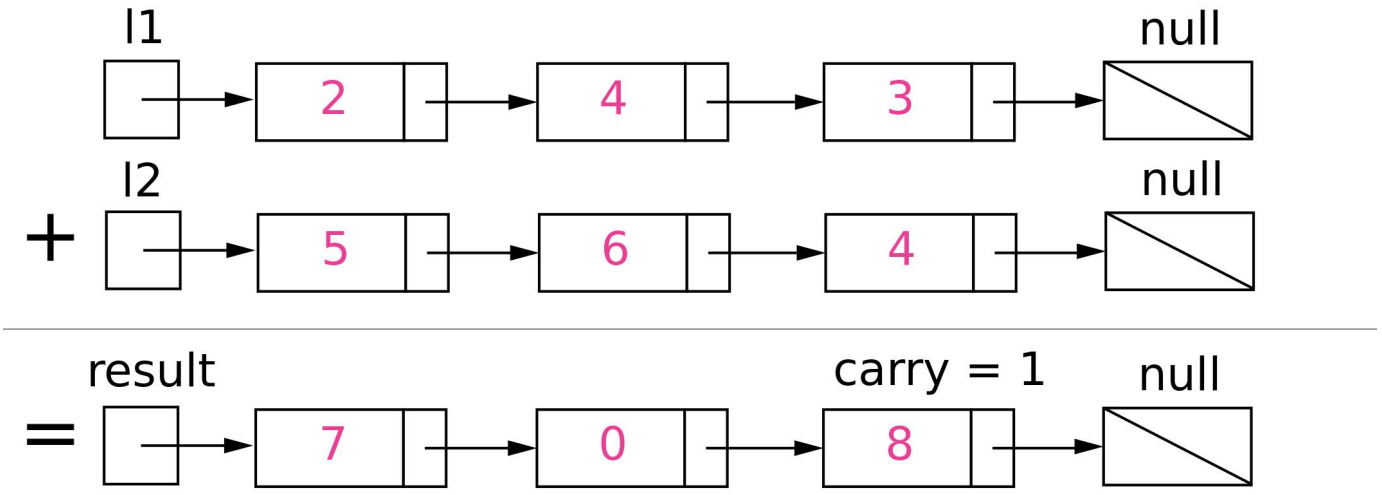
example

Input: (2 -> 4 -> 3) + (5 -> 6 -> 4)
Output: 7 -> 0 -> 8
Explanation: 342 + 465 = 807.

Solution

Intuition

Keep track of the carry using a variable and simulate digtis-by-digits sum starting from the head of list, which contains the least-significant digit.



Consider extra caution of the following cases:

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

Algorithm

We begin by summing the least-significant digits, which is the head of l1 and l2. Summing two digits might "overflow" and we would bring over the carry $= 1$ to the next iteration.
The pseudocode is as following:

- Initialize current node to dummy head of the returning list
- Initialize carry to 0.
- Initialize p and q to head of $l1$ and $l2$ respectively.
- Loop through lists l1 and l2 until you reach both ends.
  - Set x to node p's value. If p has reached the end of $l1$, set to 0.
  - Set y to node q's value. If q has reached the end of $l2$, set to 0.
  - Set sum $=$ x $+$ y $+$ carry.
  - Update carry $=$ sum / 10.
  - Create a new node with the digit value of (sum mod 10) and set it to current node's next, then advance current node to next.
  - Advance both p and q.
- Check if carry $= 1$, if so append a new node with digit 1 to the returning list.
- Return dummy head's next node.

Code

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;
}
```

Python

```
# Definition for singly-linked list.
# class ListNode(object):
#     def __init__(self, x):
#         self.val = x
#         self.next = None

class Solution(object):
    def addTwoNumbers(self, l1, l2):
        """
        :type l1: ListNode
```

```
        :type l2: ListNode
        :rtype: ListNode
        """
        dummyHead = ListNode(0)
        p = l1
        q = l2
        curr = dummyHead
        carry = 0
        while(p != None or q != None):
            x = p.val if p != None else 0
            y = q.val if q != None else 0
            SUM = carry + x + y
            carry = SUM // 10
            curr.next = ListNode(SUM % 10)
            curr = curr.next
            if (p != None):
                p = p.next
            if (q != None):
                q = q.next
        if (carry > 0):
            curr.next = ListNode(carry)

        return dummyHead.next
```