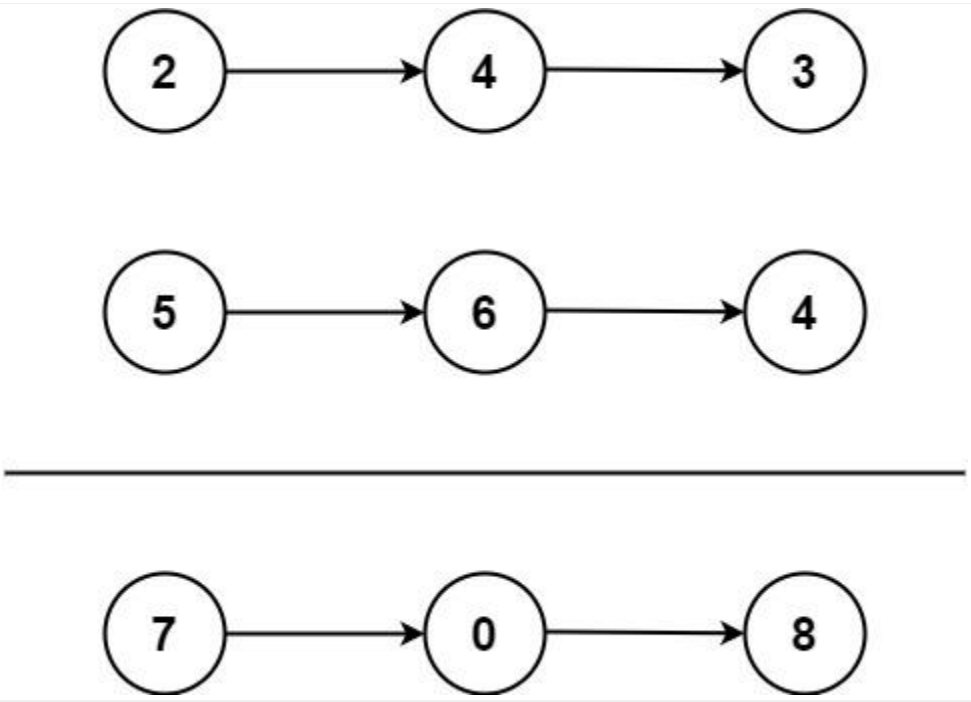
You are given two non-empty linked lists representing two non-negative integers. The digits are stored in reverse order, and each of their nodes contains a single digit. Add the two numbers and return the sum as a linked list.

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

Example 1:



Input: l1 = [2,4,3], l2 = [5,6,4]
Output: [7,0,8]
Explanation: 342 + 465 = 807.
Example 2:

Input: l1 = [0], l2 = [0]
Output: [0]
Example 3:

Input: l1 = [9,9,9,9,9,9,9], l2 = [9,9,9,9]
Output: [8,9,9,9,0,0,0,1]
Constraints:

- **The number of nodes in each linked list is in the range [1, 100].**
- **0 <= Node.val <= 9**
- **It is guaranteed that the list represents a number that does not have leading zeros.**

Solution:

```
class Solution {
   public ListNode addTwoNumbers(ListNode l1, ListNode l2) {
       int carry=0;
      ListNode head=new ListNode(0);
      ListNode p1=l1,p2=l2,p3=head;
      while(p1!=null||p2!=null){
         if(p1!=null){
            carry+=p1.val;
            p1=p1.next;
         }
          if(p2!=null){
            carry+=p2.val;
            p2=p2.next;
         }
         p3.next=new ListNode(carry%10);
         p3=p3.next;
         carry/=10;
      }
      if(carry==1)
          p3.next=new ListNode(1);
       return head.next;
   }
}
```