Subtract two LL

```
class Node:
    def __init__(self, val):
        self.data = val
        self.next = None


def subtractLinkedList(l1, l2):
    # Code here
    # return head of difference list
    len1 = length(l1)
    len2 = length(l2)
    l1 = reverse(l1)
    l2 = reverse(l2)
    if len1 > len2:
        head = subtract(l1, l2)
        head = reverse(head)
    else:
        head = subtract(l2, l1)
        head = reverse(head)
    return head


def subtract(l1, l2):
    dummyNode = Node(-1)
    prev = dummyNode
    curr1 = l1
    curr2 = l2
    carry = 0
    while curr1 != None and curr2 != None:
        if curr1.data + carry < curr2.data:
            temp = carry + curr1.data - curr2.data
            temp = temp + 10
            carry = -1
            node = Node(temp)
            prev.next = node
            prev = node
        else:
            temp = carry + curr1.data - curr2.data
            carry = 0
```

```
            node = Node(temp)
            prev.next = node
            prev = node
        curr1 = curr1.next
        curr2 = curr2.next
    while curr1 != None:
        temp = carry + curr1.data
        carry = 0
        node = Node(temp)
        prev.next = node
        prev = node
        curr1 = curr1.next
    return dummyNode.next


def reverse(head):
    curr = head
    prev = None
    forward = None
    while curr != None:
        forward = curr.next
        curr.next = prev
        prev = curr
        curr = forward
    return prev


def length(head):
    count = 0
    curr = head
    while curr != None:
        curr = curr.next
        count += 1
    return count


head1 = Node(9)
# head1.next = Node(2)
# head1.next.next = Node(3)
# head1.next.next.next = Node(4)
# head1.next.next.next.next = Node(5)
# head1.next.next.next.next.next = Node(6)
# head1.next.next.next.next.next.next = Node(7)
```

```
head2 = Node(7)
head2.next = Node(8)
head2.next.next = Node(9)

head = subtractLinkedList(head1, head2)

while head!=None:
    print(head.data,end='')
    head = head.next
```