```
        }
        cur = head;
        Node curCopy = null;
        // 设置复制节点的 rand 指针
        while (cur != null) {
                next = cur.next.next;
                curCopy = cur.next;
                curCopy.rand = cur.rand != null ? cur.rand.next : null;
                cur = next;
        }
        Node res = head.next;
        cur = head;
        // 拆分
        while (cur != null) {
                next = cur.next.next;
                curCopy = cur.next;
                cur.next = next;
                curCopy.next = next != null ? next.next : null;
                cur = next;
        }
        return res;
}
```

两个单链表生成相加链表

【题目】

假设链表中每一个节点的值都在 0～9 之间，那么链表整体就可以代表一个整数。

例如：9->3->7，可以代表整数 937。

给定两个这种链表的头节点 head1 和 head2，请生成代表两个整数相加值的结果链表。

例如：链表 1 为 9->3->7，链表 2 为 6->3，最后生成新的结果链表为 1->0->0->0。

【难度】

士 ★☆☆☆

【解答】

这道题难度较低，考查面试者基本的代码实现能力。一种实现方式是将两个链表先算出各自所代表的整数，然后求出两个整数的和，最后将这个和转换成链表的形式，但是这种方法有一个很大的问题，链表的长度可以很长，可以表达一个很大的整数，因此转成系统中的 int 类型时可能会溢出，所以不推荐这种方法。

方法一：利用栈结构求解。

1．将两个链表分别从左到右遍历，遍历过程中将值压栈，这样就生成了两个链表节点值的逆序栈，分别表示为 s1 和 s2。

例如：链表 9->3->7，s1 从栈顶到栈底为 7，3，9；链表 6->3，s2 从栈顶到栈底为 3，6。

2．将 s1 和 s2 同步弹出，这样就相当于两个链表从低位到高位依次弹出，在这个过程中生成相加链表即可，同时需要关注每一步是否有进位，用 ca 表示。

例如：s1 先弹出 7，s2 先弹出 3，这一步相加结果为 10，产生了进位，令 ca=1，然后生成一个节点值为 0 的新节点，记为 new1；s1 再弹出 3，s2 再弹出 6，这时进位为 ca=1，所以这一步相加结果为 10，继续产生进位，仍令 ca=1，然后生成一个节点值为 0 的新节点记为 new2，令 new2.next=new1；s1 再弹出 9，s2 为空，这时 ca=1，这一步相加结果为 10，仍令 ca=1，然后生成一个节点值为 0 的新节点，记为 new3，令 new3.next=new2。这一步也是模拟简单的从低位到高位进位相加的过程。

3．当 s1 和 s2 都为空时，还要关注一下进位信息是否为 1，如果为 1，比如步骤 2 中的例子，表示还要生成一个节点值为 1 的新节点，记为 new4，令 new4.next=new3。

4．返回新生成的结果链表即可。

具体过程请参看如下代码中的 addLists1 方法。

```
public class Node {
        public int value;
        public Node next;

        public Node(int data) {
                this.value = data;
        }
}

public Node addLists1(Node head1, Node head2) {
        Stack<Integer> s1 = new Stack<Integer>();
        Stack<Integer> s2 = new Stack<Integer>();
        while (head1 != null) {
                s1.push(head1.value);
                head1 = head1.next;
        }
        while (head2 != null) {
                s2.push(head2.value);
                head2 = head2.next;
        }
        int ca = 0;
        int n1 = 0;
        int n2 = 0;
        int n = 0;
```

```
        Node node = null;
        Node pre = null;
        while (!s1.isEmpty() || !s2.isEmpty()) {
                n1 = s1.isEmpty() ? 0 : s1.pop();
                n2 = s2.isEmpty() ? 0 : s2.pop();
                n = n1 + n2 + ca;
                pre = node;
                node = new Node(n % 10);
                node.next = pre;
                ca = n / 10;
        }
        if (ca == 1) {
                pre = node;
                node = new Node(1);
                node.next = pre;
        }
        return node;
}
```

方法二：利用链表的逆序求解，可以省掉用栈的空间。

1．将两个链表逆序，这样就可以依次得到从低位到高位的数字。

例如：链表 9->3->7，逆序后变为 7->3->9；链表 6->3，逆序后变为 3->6。

2．同步遍历两个逆序后的链表，这样就依次得到两个链表从低位到高位的数字，在这个过程中生成相加链表即可，同时需要关注每一步是否有进位，用 ca 表示。具体过程与方法一的步骤 2 相同。

3．当两个链表都遍历完成后，还要关注进位信息是否为 1，如果为 1，还要生成一个节点值为 1 的新节点。

4．将两个逆序的链表再逆序一次，即调整成原来的样子。

5．返回新生成的结果链表。

具体过程请参看如下代码中的 addLists2 方法。

```
public Node addLists2(Node head1, Node head2) {
        head1 = reverseList(head1);
        head2 = reverseList(head2);
        int ca = 0;
        int n1 = 0;
        int n2 = 0;
        int n = 0;
        Node c1 = head1;
        Node c2 = head2;
        Node node = null;
        Node pre = null;
        while (c1 != null || c2 != null) {
                n1 = c1 != null ? c1.value : 0;
                n2 = c2 != null ? c2.value : 0;
```

```
                n = n1 + n2 + ca;
                pre = node;
                node = new Node(n % 10);
                node.next = pre;
                ca = n / 10;
                c1 = c1 != null ? c1.next : null;
                c2 = c2 != null ? c2.next : null;
        }
        if (ca == 1) {
                pre = node;
                node = new Node(1);
                node.next = pre;
        }
        reverseList(head1);
        reverseList(head2);
        return node;
}

public Node reverseList(Node head) {
        Node pre = null;
        Node next = null;
        while (head != null) {
                next = head.next;
                head.next = pre;
                pre = head;
                head = next;
        }
        return pre;
}
```

两个单链表相交的一系列问题

【题目】

在本题中，单链表可能有环，也可能无环。给定两个单链表的头节点 head1 和 head2，这两个链表可能相交，也可能不相交。请实现一个函数，如果两个链表相交，请返回相交的第一个节点；如果不相交，返回 null 即可。

要求：如果链表 1 的长度为 N，链表 2 的长度为 M，时间复杂度请达到 $O(N+M)$，额外空间复杂度请达到 $O(1)$。

【难度】

将 ★★★★