```
        }
        // 所有的重新连接
        if (eT != null) {
                eT.next = bH;
        }
        return sH != null ? sH : eH != null ? eH : bH;
}
```

复制含有随机指针节点的链表

【题目】

一种特殊的链表节点类描述如下：

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

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

Node 类中的 value 是节点值，next 指针和正常单链表中 next 指针的意义一样，都指向下一个节点，rand 指针是 Node 类中新增的指针，这个指针可能指向链表中的任意一个节点，也可能指向 null。

给定一个由 Node 节点类型组成的无环单链表的头节点 head，请实现一个函数完成这个链表中所有结构的复制，并返回复制的新链表的头节点。例如：链表 1->2->3->null，假设 1 的 rand 指针指向 3，2 的 rand 指针指向 null，3 的 rand 指针指向 1。复制后的链表应该也是这种结构，比如，1′->2′->3′->null，1′的 rand 指针指向 3′，2′的 rand 指针指向 null，3′的 rand 指针指向 1′，最后返回 1′。

进阶：不使用额外的数据结构，只用有限几个变量，且在时间复杂度为 $O(N)$内完成原问题要实现的函数。

【难度】

尉 ★★☆☆

【解答】

首先介绍普通解法，普通解法可以做到时间复杂度为 $O(N)$，额外空间复杂度为 $O(N)$，需要使用到哈希表（HashMap）结构。具体过程如下：

1．首先从左到右遍历链表，对每个节点都复制生成相应的副本节点，然后将对应关系放入哈希表 map 中。例如，链表 1->2->3->null，遍历 1、2、3 时依次生成 1′、2′、3′，最后将对应关系放入 map 中：

key	value	意义
1	1′	表示节点 1 复制了节点 1′
2	2′	表示节点 2 复制了节点 2′
3	3′	表示节点 3 复制了节点 3′

步骤 1 完成后，原链表没有任何变化，每一个副本节点的 next 和 rand 指针都指向 null。

2．再从左到右遍历链表，此时就可以设置每一个副本节点的 next 和 rand 指针。

例如：原链表 1->2->3->null，假设 1 的 rand 指针指向 3，2 的 rand 指针指向 null，3 的 rand 指针指向 1。遍历到节点 1 时，可以从 map 中得到节点 1 的副本节点 1′，节点 1 的 next 指向节点 2，所以从 map 中得到节点 2 的副本节点 2′，然后令 1′.next=2′，副本节点 1′的 next 指针就设置好了。同时节点 1 的 rand 指向节点 3，所以从 map 中得到节点 3 的副本节点 3′，然后令 1′.rand=3′，副本节点 1′的 rand 指针也设置好了。以这种方式可以设置每一个副本节点的 next 与 rand 指针。

3．将 1′节点作为结果返回即可。

哈希表增删改查的操作时间复杂度都是 $O(1)$，普通方法一共只遍历链表两遍，所以普通解法的时间复杂度为 $O(N)$，因为使用了哈希表来保存原节点与副本节点的对应关系，所以额外空间复杂度为 $O(N)$。

具体过程请参看如下代码中的 copyListWithRand1 方法。

```
public Node copyListWithRand1(Node head) {
        HashMap<Node, Node> map = new HashMap<Node, Node>();
        Node cur = head;
        while (cur != null) {
                map.put(cur, new Node(cur.value));
                cur = cur.next;
        }
        cur = head;
        while (cur != null) {
                map.get(cur).next = map.get(cur.next);
                map.get(cur).rand = map.get(cur.rand);
```

```
                cur = cur.next;
            }
            return map.get(head);
        }
```

接下来介绍进阶解法，进阶解法不使用哈希表来保存对应关系，而只用有限的几个变量完成所有的功能。具体过程如下：

1．首先从左到右遍历链表，对每个节点 cur 都复制生成相应的副本节点 copy，然后把 copy 放在 cur 和下一个要遍历节点的中间。

例如：原链表 1->2->3->null，在步骤 1 中完成后，原链表变成 1->1′->2->2′->3->3′->null。

2．再从左到右遍历链表，在遍历时设置每一个副本节点的 rand 指针。还是举例来说明调整过程。

例如：此时链表为 1->1′->2->2′->3->3′->null，假设 1 的 rand 指针指向 3，2 的 rand 指针指向 null，3 的 rand 指针指向 1。遍历到节点 1 时，节点 1 的下一个节点 1.next 就是其副本节点 1′。1 的 rand 指针指向 3，所以 1′的 rand 指针应该指向 3′。如何找到 3′呢？因为每个节点的副本节点都在自己的后一个，所以此时通过 3.next 就可以找到 3′，令 1′.next=3′即可。以这种方式可以设置每一个副本节点的 rand 指针。

3．步骤 2 完成后，节点 1，2，3，……之间的 rand 关系没有任何变化，节点 1′，2′，3′……之间的 rand 关系也被正确设置了，此时所有的节点与副本节点串在一起，将其分离出来即可。

例如：此时链表为 1->1′->2->2′->3->3′->null，分离成 1->2->3->null 和 1′->2′->3′->null 即可。并且在这一步中，每个节点的 rand 指针不用做任何调整，在步骤 2 中都已经设置好。

4．将 1′节点作为结果返回即可。

进阶解法考查的依然是利用有限几个变量完成链表调整的代码实现能力。具体过程请参看如下代码中的 copyListWithRand2 方法。

```
public Node copyListWithRand2(Node head) {
        if (head == null) {
                return null;
        }
        Node cur = head;
        Node next = null;
        // 复制并链接每一个节点
        while (cur != null) {
                next = cur.next;
                cur.next = new Node(cur.value);
                cur.next.next = next;
                cur = next;
```

```
        }
        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。