```
                }
            }
            pre.next = cur1 == null ? cur2 : cur1;
            return head;
        }
```

按照左右半区的方式重新组合单链表

【题目】

给定一个单链表的头部节点 head，链表长度为 N，如果 N 为偶数，那么前 $N/2$ 个节点算作左半区，后 $N/2$ 个节点算作右半区；如果 N 为奇数，那么前 $N/2$ 个节点算作左半区，后 $N/2+1$ 个节点算作右半区。左半区从左到右依次记为 L1->L2->…，右半区从左到右依次记为 R1->R2->…，请将单链表调整成 L1->R1->L2->R2->…的形式。

例如：

1->null，调整为 1->null。

1->2->null，调整为 1->2->null。

1->2->3->null，调整为 1->2->3->null。

1->2->3->4->null，调整为 1->3->2->4->null。

1->2->3->4->5->null，调整为 1->3->2->4->5->null。

1->2->3->4->5->6->null，调整为 1->4->2->5->3->6->null。

【难度】

士 ★☆☆☆

【解答】

假设链表的长度为 N，直接给出时间复杂度为 $O(N)$、额外空间复杂度为 $O(1)$的方法。具体过程如下：

1．如果链表为空或长度为 1，不用调整，过程直接结束。

2．链表长度大于 1 时，遍历一遍找到左半区的最后一个节点，记为 mid。

例如：1->2，mid 为 1；1->2->3，mid 为 1；1->2->3->4，mid 为 2；1->2->3->4->5，mid 为 2；1->2->3->4->5->6，mid 为 3。也就是说，从长度为 2 开始，长度每增加 2，mid 就往后移动一个节点。

3．遍历一遍找到 mid 之后，将左半区与右半区分离成两个链表（mid.next=null），分别记为 left(head)和 right（原来的 mid.next）。

4．将两个链表按照题目要求合并起来。

具体过程请参看如下代码中的 relocate 方法，其中的 mergeLR 方法为步骤 4 的合并过程。

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

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

public void relocate(Node head) {
        if (head == null || head.next == null) {
                return;
        }
        Node mid = head;
        Node right = head.next;
        while (right.next != null && right.next.next != null) {
                mid = mid.next;
                right = right.next.next;
        }
        right = mid.next;
        mid.next = null;
        mergeLR(head, right);
}

public void mergeLR(Node left, Node right) {
        Node next = null;
        while (left.next != null) {
                next = right.next;
                right.next = left.next;
                left.next = right;
                left = right.next;
                right = next;
        }
        left.next = right;
}
```