

```
        if (cur.value < small.value) {  
            smallPre = pre;  
            small = cur;  
        }  
        pre = cur;  
        cur = cur.next;  
    }  
    return smallPre;  
}
```

一种怪异的节点删除方式

【题目】

链表节点值类型为 `int` 型，给定一个链表中的节点 `node`，但不给定整个链表的头节点。如何在链表中删除 `node`？请实现这个函数，并分析这么会出现哪些问题。

要求：时间复杂度为 $O(1)$ 。

【难度】

士 ★☆☆☆

【解答】

本题的思路很简单，举例就能说明具体的做法。

例如，链表 `1->2->3->null`，只知道要删除节点 2，而不知道头节点。那么只需把节点 2 的值变成节点 3 的值，然后在链表中删除节点 3 即可。

这道题目出现的次数很多，这么做看起来非常方便，但其实是有很大问题的。

问题一：这样的删除方式无法删除最后一个节点。还是以原示例来说明，如果知道要删除节点 3，而不知道头节点。但它是最后的节点，根本没有下一个节点来代替节点 3 被删除，那么只有让节点 2 的 `next` 指向 `null` 这一种办法，而我们又根本找不到节点 2，所以根本没法正确删除节点 3。读者可能会问，我们能不能把节点 3 在内存上的区域变成 `null` 呢？这样不就相当于让节点 2 的 `next` 指针指向了 `null`，起到节点 3 被删除的效果了吗？不可以。`null` 在系统中是一个特定的区域，如果想让节点 2 的 `next` 指针指向 `null`，必须找到节点 2。

问题二：这种删除方式在本质上根本就不是删除了 `node` 节点，而是把 `node` 节点的值改变，然后删除 `node` 的下一个节点，在实际的工程中可能会带来很大问题。比如，工程上

的一个节点可能代表很复杂的结构，节点值的复制会相当复杂，或者可能改变节点值这个操作都是被禁止的；再如，工程上的一个节点代表提供服务的一个服务器，外界对每个节点都有很多依赖，比如，示例中删除节点 2 时，其实影响了节点 3 对外提供的服务。

这种删除方式的具体过程请参看如下代码中的 `removeNodeWired` 方法。

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

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

public void removeNodeWired(Node node) {
    if (node == null) {
        return;
    }
    Node next = node.next;
    if (next == null) {
        throw new RuntimeException("can not remove last node.");
    }
    node.value = next.value;
    node.next = next.next;
}
```

向有序的环形单链表中插入新节点

【题目】

一个环形单链表从头节点 `head` 开始不降序，同时由最后的节点指回头节点。给定这样一个环形单链表的头节点 `head` 和一个整数 `num`，请生成节点值为 `num` 的新节点，并插入到这个环形链表中，保证调整后的链表依然有序。

【难度】

士 ★☆☆☆

【解答】

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. 生成节点值为 `num` 的新节点，记为 `node`。