```
    }

    public void printCommonPart(Node head1, Node head2) {
        System.out.print("Common Part: ");
        while (head1 != null && head2 != null) {
            if (head1.value < head2.value) {
                head1 = head1.next;
            } else if (head1.value > head2.value) {
                head2 = head2.next;
            } else {
                System.out.print(head1.value + " ");
                head1 = head1.next;
                head2 = head2.next;
            }
        }
        System.out.println();
    }
```

在单链表和双链表中删除倒数第 K 个节点

【题目】

分别实现两个函数，一个可以删除单链表中倒数第 K 个节点，另一个可以删除双链表中倒数第 K 个节点。

【要求】

如果链表长度为 N，时间复杂度达到 $O(N)$，额外空间复杂度达到 $O(1)$。

【难度】

士 ★☆☆☆

【解答】

本题较为简单，实现方式也是多种多样的，本书提供一种方法供读者参考。

先来看看单链表如何调整。如果链表为空或者 K 值小于 1，这种情况下，参数是无效的，直接返回即可。除此之外，让链表从头开始走到尾，每移动一步，就让 K 的值减 1。

链表：1->2->3，$K = 4$，链表根本不存在倒数第 4 个节点。

走到的节点：1 -> 2 -> 3

K 变化为：3　2　1

链表：1->2->3，$K=3$，链表倒数第 3 个节点是 1 节点。

走到的节点：1 -> 2 -> 3

K 变化为：2 1 0

链表：1->2->3，$K=2$，链表倒数第 2 个节点是 2 节点。

走到的节点：1 -> 2 -> 3

K 变化为：1 0 -1

由以上三种情况可知，让链表从头开始走到尾，每移动一步，就让 K 值减 1，当链表走到结尾时，如果 K 值大于 0，说明不用调整链表，因为链表根本没有倒数第 K 个节点，此时将原链表直接返回即可；如果 K 值等于 0，说明链表倒数第 K 个节点就是头节点，此时直接返回 head.next，也就是原链表的第二个节点，让第二个节点作为链表的头返回即可，相当于删除头节点；接下来，说明一下如果 K 值小于 0，该如何处理。

先明确一点，如果要删除链表的头节点之后的某个节点，实际上需要找到要删除节点的前一个节点，比如：1->2->3，如果想删除节点 2，则需要找到节点 1，然后把节点 1 连到节点 3 上（1->3），以此来达到删除节点 2 的目的。

如果 K 值小于 0，如何找到要删除节点的前一个节点呢？方法如下：

1．重新从头节点开始走，每移动一步，就让 K 的值加 1。

2．当 K 等于 0 时，移动停止，移动到的节点就是要删除节点的前一个节点。

这样做是非常好理解的，因为如果链表长度为 N，要删除倒数第 K 个节点，很明显，倒数第 K 个节点的前一个节点就是第 N-K 个节点。在第一次遍历后，K 的值变为 K-N。第二次遍历时，K 的值不断加 1，加到 0 就停止遍历，第二次遍历当然会停到第 N-K 个节点的位置。

具体过程请参看如下代码中的 removeLastKthNode 方法。

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

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

public Node removeLastKthNode(Node head, int lastKth) {
        if (head == null || lastKth < 1) {
                return head;
        }
        Node cur = head;
```

```
        while (cur != null) {
                lastKth--;
                cur = cur.next;
        }
        if (lastKth == 0) {
                head = head.next;
        }
        if (lastKth < 0) {
                cur = head;
                while (++lastKth != 0) {
                        cur = cur.next;
                }
                cur.next = cur.next.next;
        }
        return head;
}
```

对于双链表的调整，几乎与单链表的处理方式一样，注意 last 指针的重连即可。具体过程请参看如下代码中的 removeLastKthNode 方法。

```
public class DoubleNode {
        public int value;
        public DoubleNode last;
        public DoubleNode next;

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


public DoubleNode removeLastKthNode(DoubleNode head, int lastKth) {
        if (head == null || lastKth < 1) {
                return head;
        }
        DoubleNode cur = head;
        while (cur != null) {
                lastKth--;
                cur = cur.next;
        }
        if (lastKth == 0) {
                head = head.next;
                head.last = null;
        }
        if (lastKth < 0) {
                cur = head;
                while (++lastKth != 0) {
                        cur = cur.next;
                }
                DoubleNode newNext = cur.next.next;
                cur.next = newNext;
```

```
                if (newNext != null) {
                        newNext.last = cur;
                }
            }
            return head;
        }
```

删除链表的中间节点和 a/b 处的节点

【题目】

给定链表的头节点 head，实现删除链表的中间节点的函数。

例如：

不删除任何节点；

1->2，删除节点 1；

1->2->3，删除节点 2；

1->2->3->4，删除节点 2；

1->2->3->4->5，删除节点 3；

进阶：

给定链表的头节点 head、整数 a 和 b，实现删除位于 a/b 处节点的函数。

例如：

链表：1->2->3->4->5，假设 a/b 的值为 r。

如果 r 等于 0，不删除任何节点；

如果 r 在区间(0, 1/5]上，删除节点 1；

如果 r 在区间(1/5, 2/5]上，删除节点 2；

如果 r 在区间(2/5, 3/5]上，删除节点 3；

如果 r 在区间(3/5, 4/5]上，删除节点 4；

如果 r 在区间(4/5, 1]上，删除节点 5；

如果 r 大于 1，不删除任何节点。

【难度】

士 ★☆☆☆