如果链表长度为 7，a=5，b=6。

r = (7*5)/6 = 5.8333…，向上取整后为 6，所以应该删除第 6 个节点。

如果链表长度为 7，a=1，b=6。

r = (7*1)/6 = 1.1666…，向上取整后为 2，所以应该删除第 2 个节点。

知道该删除第几个节点之后，接下来找到需要删除节点的前一个节点即可。具体过程请参看如下代码中的 removeByRatio 方法。

```
public Node removeByRatio(Node head, int a, int b) {
        if (a < 1 || a > b) {
                return head;
        }
        int n = 0;
        Node cur = head;
        while (cur != null) {
                n++;
                cur = cur.next;
        }
        n = (int) Math.ceil(((double) (a * n)) / (double) b);
        if (n == 1) {
                head = head.next;
        }
        if (n > 1) {
                cur = head;
                while (--n != 1) {
                        cur = cur.next;
                }
                cur.next = cur.next.next;
        }
        return head;
}
```

反转单向和双向链表

【题目】

分别实现反转单向链表和反转双向链表的函数。

【要求】

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

【难度】

士 ★☆☆☆

【解答】

本题比较简单，读者做到代码一次成型，运行不出错即可。

反转单向链表的函数如下，函数返回反转之后链表新的头节点：

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

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;
}
```

反转双向链表的函数如下，函数返回反转之后链表新的头节点：

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

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