1.循环链表

1. 哈希表法

```java
    public boolean hasCycle(ListNode head) {
        HashSet<ListNode> hashSet = new HashSet<ListNode>();
        while (head != null) {
            // hashSet.add(head) 如果里面没有该节点返回true, 并且将head节点添加到
hashset里边
            // 如果里面有该节点返回false
            if (!hashSet.add(head)) return true;
            //指针指向
            head = head.next;
        }
        return false;
    }
```

2.快慢指针法

```java
    public boolean hasCycle(ListNode head) {
        if (head == null) return false;
        ListNode fast = head, slow = head;
        do {
            if (fast == null || fast.next == null) {
                return false;
            }
            slow = slow.next;
            fast = fast.next.next;
        } while (fast != slow);
        return true;
    }
```

2.循环链表||

1.哈希表

```java
    public ListNode detectCycle(ListNode head) {
        HashSet<ListNode> hashSet = new HashSet<ListNode>();
        while (head != null) {
            if (!hashSet.add(head)) {
                return head;
            }
            head = head.next;
        }
        return null;
    }
```

2.快慢指针

```java
    public ListNode detectCycle(ListNode head) {
      if (head==null )return null ;
```

```
        ListNode fast= head,slow=head;
        do {
            if (fast==null || fast.next==null) return null;
            fast=fast.next.next;
            slow=slow.next;
        }while (fast!=slow);
        // ListNode newNode=head;
          fast=head;
          while (fast!=slow){
              slow=slow.next;
              fast=fast.next;
          }
          return fast;
    }
```

3.快乐数

```
    public boolean isHappy(int n) {
        int fast = n, slow = n;
        do {
            fast = getNext(getNext(fast));
            slow = getNext(slow);
        } while (fast != slow && fast != 1);
        return fast == 1;
    }

    public int getNext(int n) {
        int sum = 0;
        while (n > 0) {
            // 15  5* 5
            sum += (n % 10) * (n % 10);
            n = n / 10;
        }
        return sum;
    }
```

4.反转链表

```
    public ListNode reverseList(ListNode head) {
        ListNode pre = null, curr = head, next = null;
        while (curr != null) {
            next = curr.next;
            curr.next = pre;
            pre = curr;
            curr = next;
        }
        return pre;
    }
```

5.反转链表II

```
public ListNode reverseBetween(ListNode head, int left, int right) {
        ListNode hair = new ListNode(0, head), con = hair, tail = null;
```

```java
        int n = right - left + 1;
        while (left > 1) {
            con = con.next;
            left--;
        }
        con.next = reverse(con.next, n);
        return hair.next;
    }

    public ListNode reverse(ListNode head, int n) {
        ListNode pre = new ListNode(), curr = head, next = null;
        while (n > 0) {
            next = curr.next;
            curr.next = pre.next;
            pre.next = curr;
            curr = next;
            n--;
        }
        head.next = curr;
        return pre.next;
    }
```

6.K个一组反转链表

```
```

7.旋转链表

```
```

8.两两交换链表的节点

```
```

9.删除链表的倒数第N个节点

```
```

10.删除排序链表中的重复元素

```
```

11.删除排序链表中的重复元素II

```
```