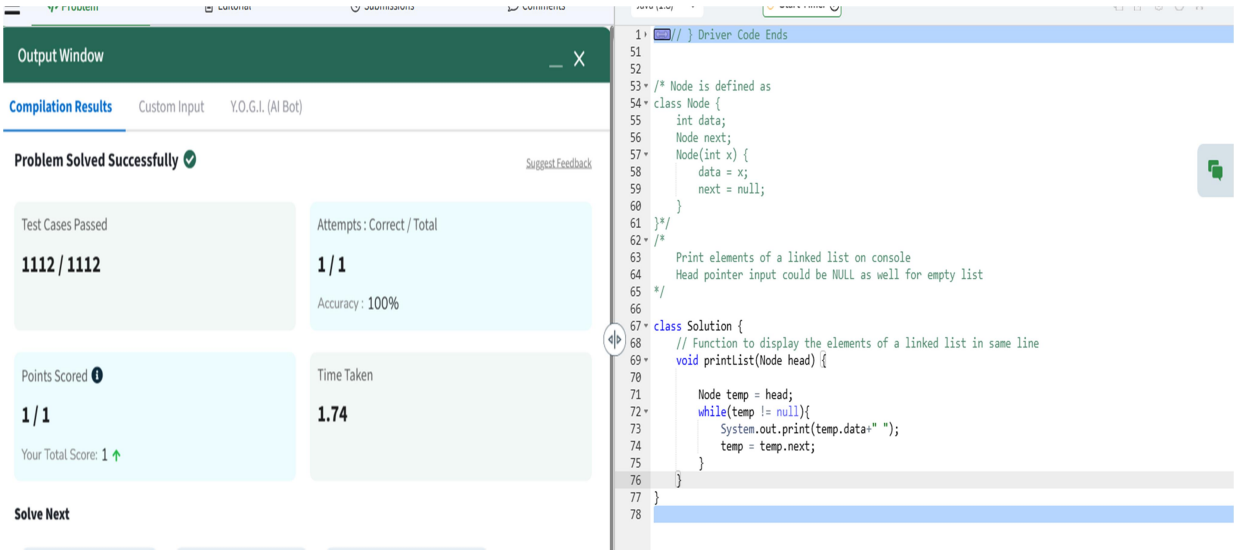
Assignment – 3

1. **PrintLinked list**

Code:

```
class Solution {
    void printList(Node head) {

        Node temp = head;
        while(temp != null){
            System.out.print(temp.data+" ");
            temp = temp.next;
        }
    }
}
```

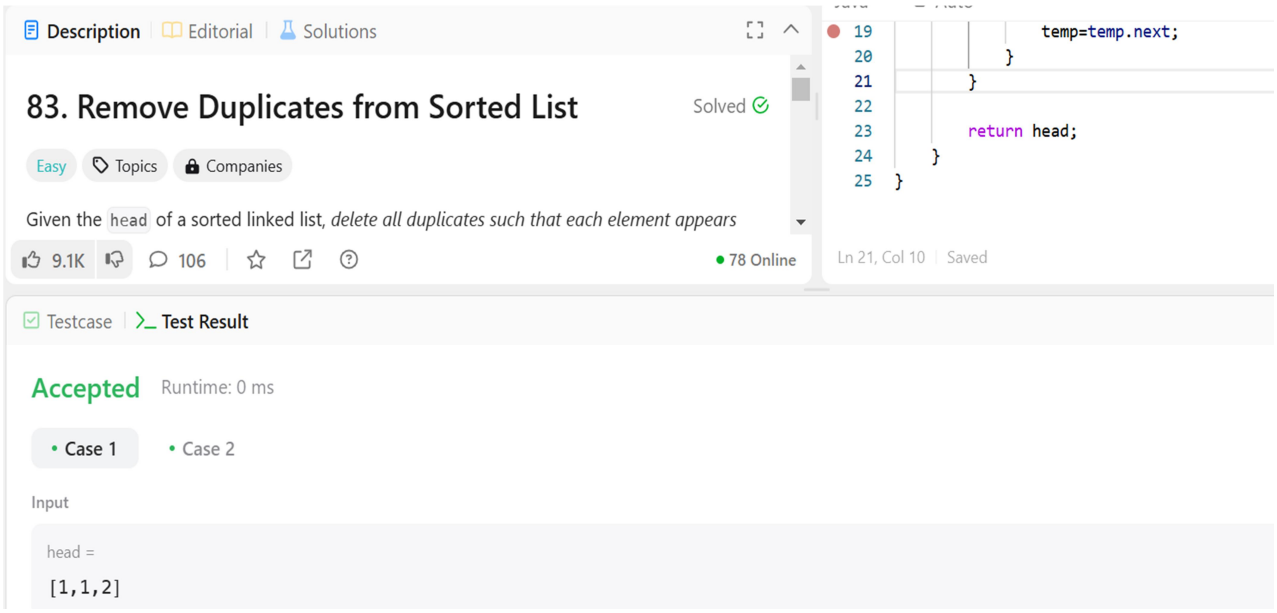


2. **Remove duplicates from sorted list**

Code:

```
class Solution {
    public ListNode deleteDuplicates(ListNode head) {
        ListNode temp=head;
        while(temp!=null&&temp.next!=null){
```

```
            if(temp.val==temp.next.val){
                temp.next=temp.next.next;
            }
            else{
                temp=temp.next;
            }
        }
        return head;
    }
}
```

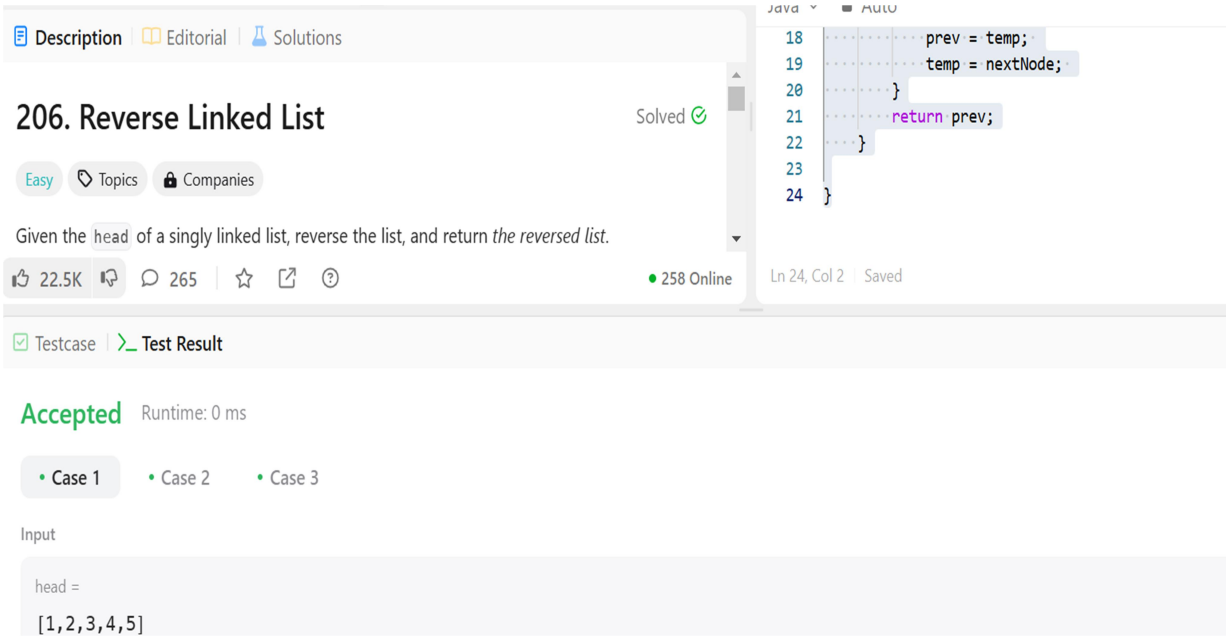


3. Reverse a LinkedList

Code:

```
class Solution {
    public ListNode reverseList(ListNode head) {
        ListNode prev = null;
        ListNode temp = head;
        while (temp != null) {
```

```
            ListNode nextNode = temp.next;
            temp.next = prev;
            prev = temp;
            temp = nextNode;
        }
        return prev;
    }
}
```

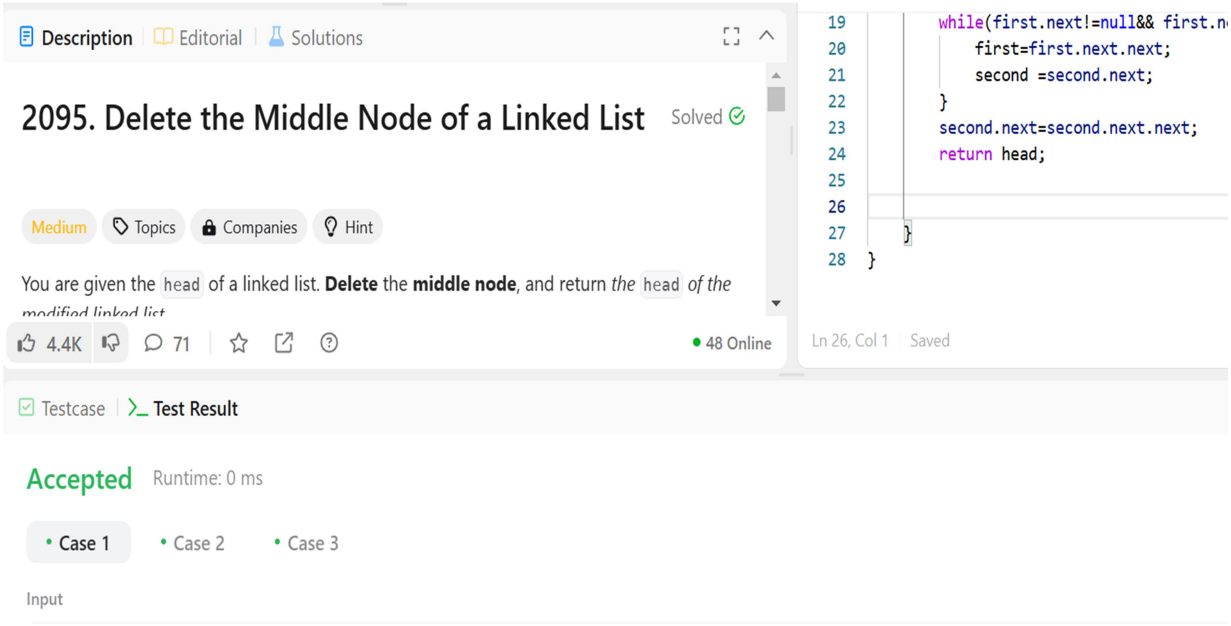


4. Delete Middle node of a list

Code:

```
class Solution {
    public ListNode deleteMiddle(ListNode head) {
        if(head.next==null){
            return null;
```

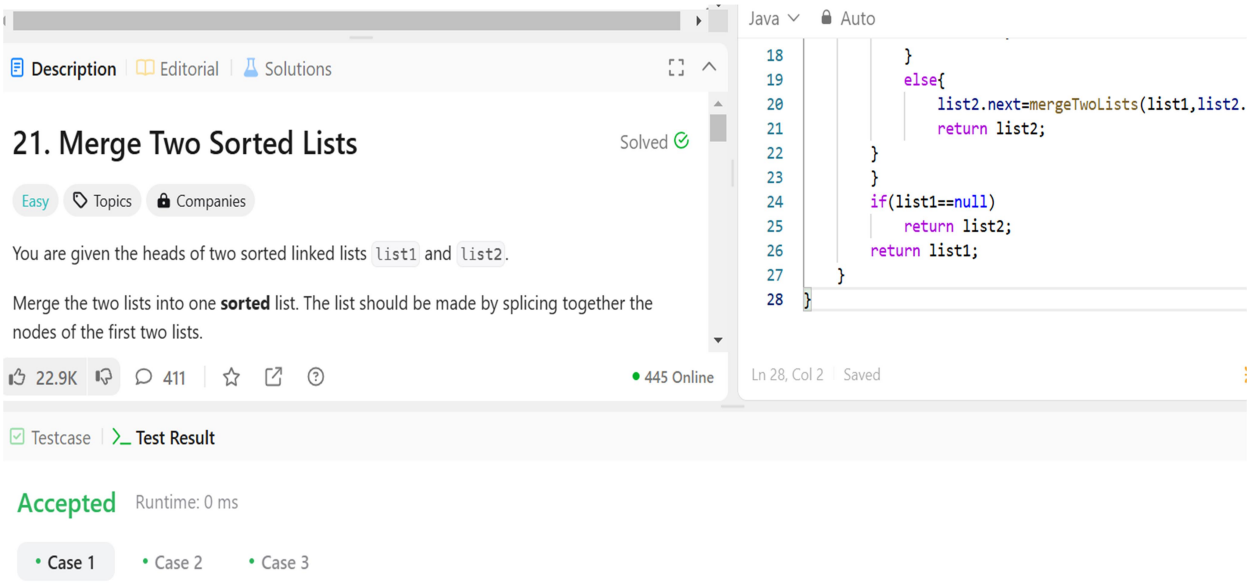
```
        }
        ListNode first=head.next;
        ListNode second=head;

        while(first.next!=null&& first.next.next!=null){
            first=first.next.next;
            second =second.next;
        }
        second.next=second.next.next;
        return head;
    }
}
```



5. Merge two sorted linked lists

Code:

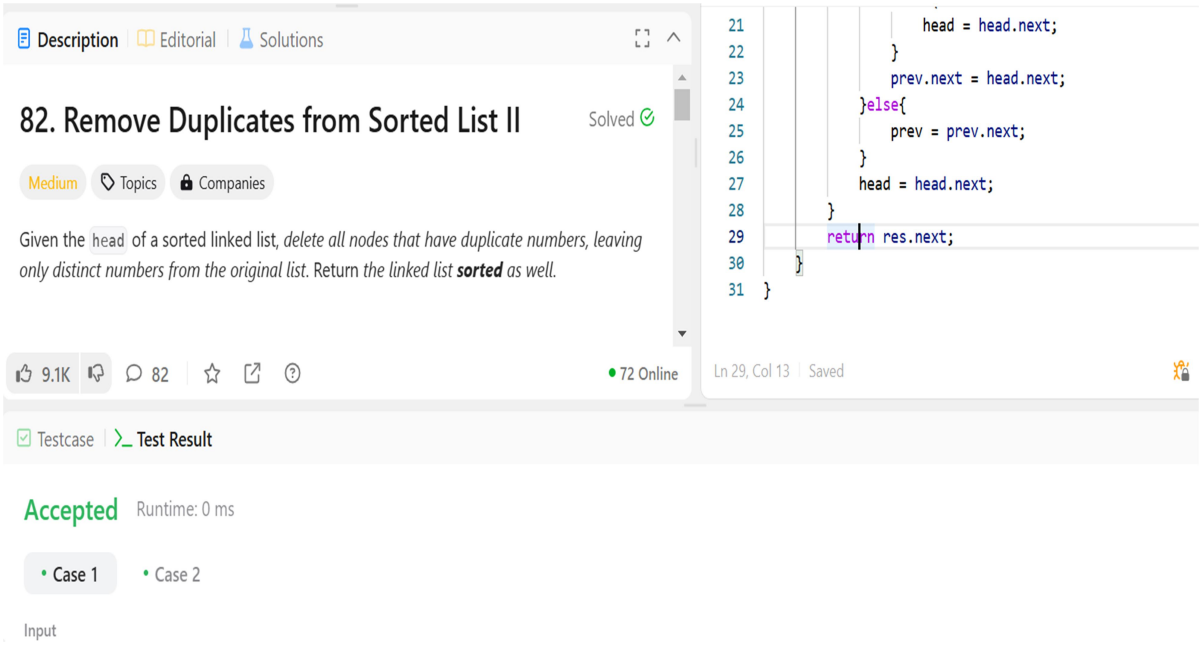
```
class Solution {
    public ListNode mergeTwoLists(ListNode list1, ListNode list2) {

        if(list1!=null && list2!=null){
        if(list1.val<list2.val){
            list1.next=mergeTwoLists(list1.next,list2);
            return list1;
            }
            else{
                list2.next=mergeTwoLists(list1,list2.next);
                return list2;
        }
        }
        if(list1==null)
            return list2;
        return list1;
    }
}
```



6. Remove duplicates from sorted lists 2

Code:

```
class Solution {
    public ListNode deleteDuplicates(ListNode head) {
        if (head == null || head.next == null) return head;
        ListNode res = new ListNode(0,head);
        ListNode prev = res;
        while(head != null && head.next != null){
            if(head.next.val == head.val){
                while(head.next != null && head.next.val == head.val){
                    head = head.next;
                }
                prev.next = head.next;
            }else{
                prev = prev.next;
            }
            head = head.next;
        }
        return res.next;
    }
}
```

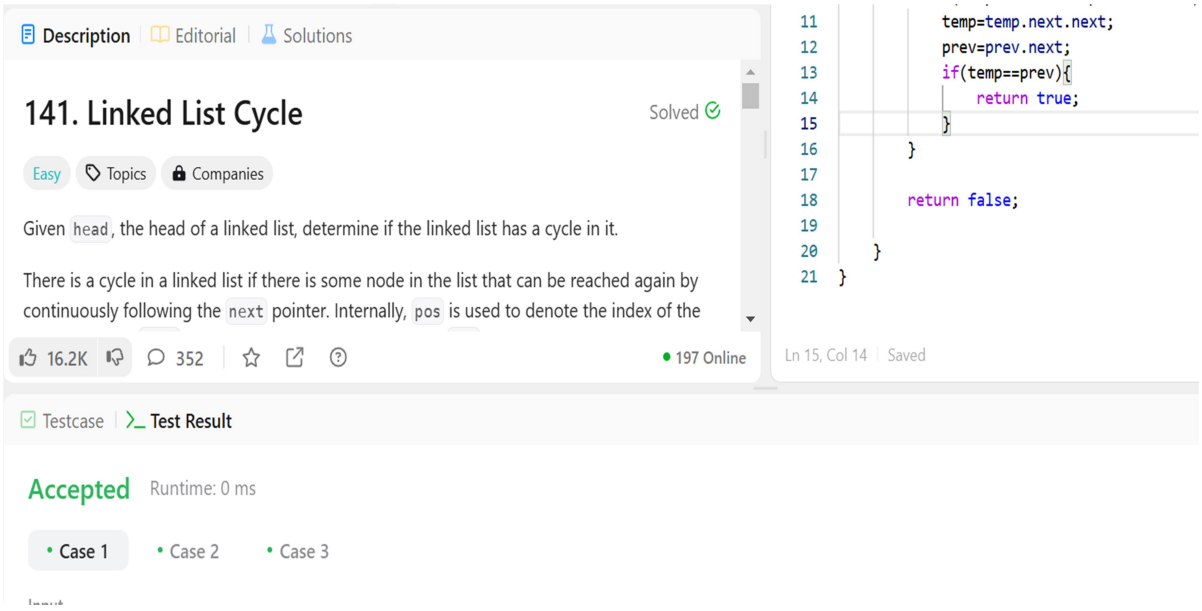


7. Detect a cycle in a linked list

Code:

```
public class Solution {
    public boolean hasCycle(ListNode head) {
        if(head==null|| head.next==null){
            return false;
        }
        ListNode temp=head;
        ListNode prev=head;
        while(temp!=null&&temp.next!=null){
            temp=temp.next.next;
            prev=prev.next;
            if(temp==prev){
```

```
            return true;
          }
        }
        return false;
    }
}
```

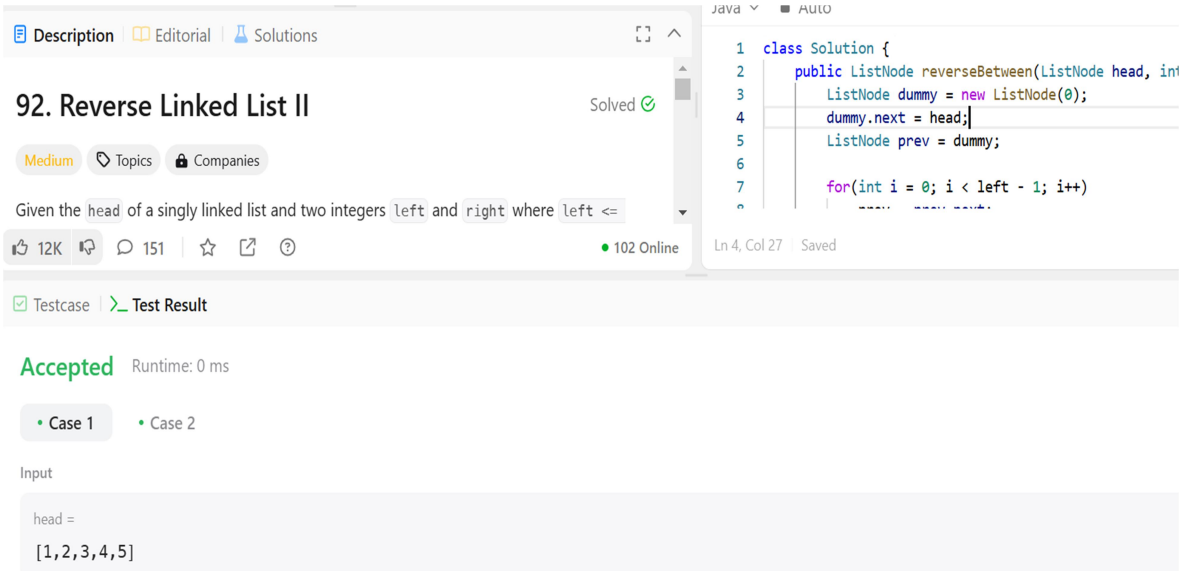


8. Reverse linked list 2

Code:

```
class Solution {
    public ListNode reverseBetween(ListNode head, int left, int right) {
        ListNode dummy = new ListNode(0);
        dummy.next = head;
        ListNode prev = dummy;

        for(int i = 0; i < left - 1; i++)
```

```
            prev = prev.next;


        ListNode curr = prev.next;
        for(int i = 0; i < right - left; i++){
            ListNode forw = curr.next;
            curr.next = forw.next;
            forw.next = prev.next;
            prev.next = forw;
        }
        return dummy.next;
    }
}
```

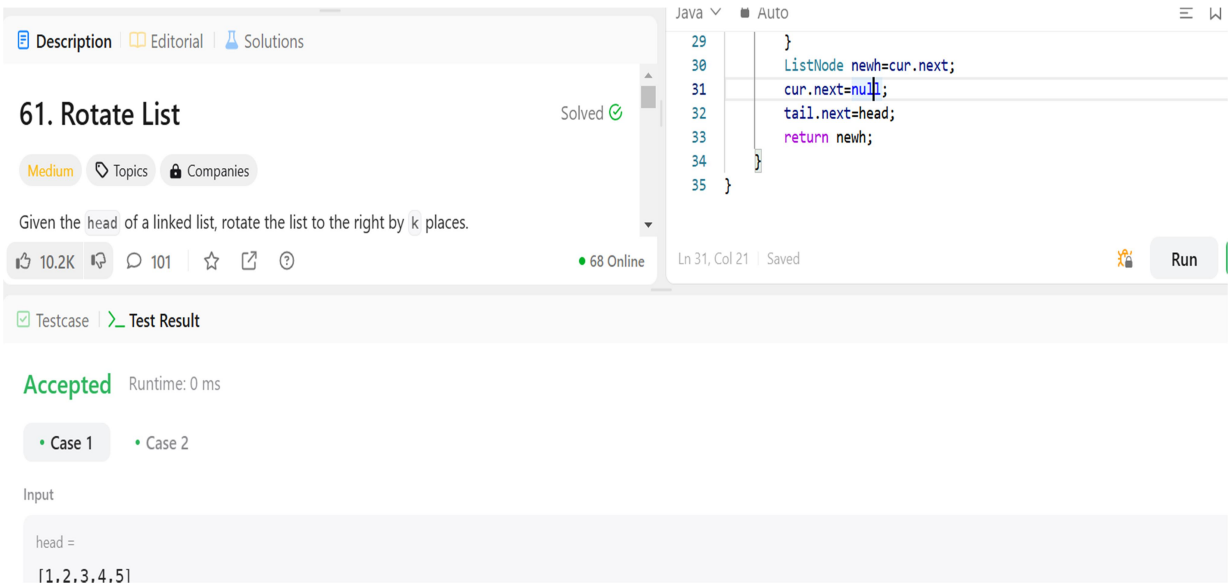


9. rotate a list

Code:

```
class Solution {
    public ListNode rotateRight(ListNode head, int k) {
```

```
        if (head==null){
            return head;
        }
        int length=1;
        ListNode tail=head;
        ListNode cur=head;
        while (tail.next!=null){
            tail=tail.next;
            length++;
        }
        k=k%length;
        if (k==0){
            return head;
        }
        for(int i=0;i<length-k-1;i++){
            cur=cur.next;
        }
        ListNode newh=cur.next;
        cur.next=null;
        tail.next=head;
        return newh;
    }
}
```



10. Sort List

Code:

```
class Solution {
    public ListNode sortList(ListNode head) {
        if(head==null || head.next==null){
            return head;
        }

        ListNode middle= middleNode(head);
        ListNode start2= middle.next;
        middle.next=null;
        ListNode first = sortList(head);
        ListNode second = sortList(start2);
```

```
return mergeTwoLists(first,second);
    }
    public ListNode middleNode(ListNode head) {
        ListNode slow=head;
        ListNode fast=head;
        while(fast!=null &&fast.next!=null && fast.next.next!=null){
            slow=slow.next;
            fast=fast.next.next;
        }
        return slow;
    }
public ListNode mergeTwoLists(ListNode list1, ListNode list2) {
        ListNode list3 = new ListNode();
        ListNode temp3 = list3;
        while (list1 != null && list2 != null) {
          if (list1.val <= list2.val) {
              temp3.next = list1;
              list1 = list1.next;
           } else {
              temp3.next = list2;
              list2 = list2.next;
           }
```

```
            temp3 = temp3.next;
        }
        if (list1 != null) {
            temp3.next = list1;
        } else {
            temp3.next = list2;
        }
        return list3.next;
    }
}
```

11. Detect a cycle in a linked list 2

Code:

```
public class Solution {
    public ListNode detectCycle(ListNode head) {
```

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