876. Middle of the Linked List

```
class Solution {
    public ListNode middleNode(ListNode head) {
      ListNode slow=head;
        ListNode fast=head;
        while (fast!=null && fast.next!=null){
            slow=slow.next;
            fast=fast.next.next;
        }
        return slow;
    }
}
```

141. Linked List Cycle

```
public class Solution {
    public boolean hasCycle(ListNode head) {
        ListNode fast= head;
        ListNode slow= head;

        while(fast !=null && fast.next != null) {
            fast = fast.next.next;
            slow= slow.next;

            if(fast == slow) {
                return true;
            }
        }
        return false;
    }       }
```

142. Linked List Cycle II

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

237. Delete Node in a Linked List

```
class Solution {
    public void deleteNode(ListNode node) {
        node.val=node.next.val;
        node.next=node.next.next;
    }
}
```

206. Reverse Linked List

```
class Solution {
    public ListNode reverseList(ListNode head) {
    ListNode prev=null;
       ListNode current=head;
       while(current !=null){
        ListNode next=current.next;
        current.next=prev;
        prev=current;
        current=next;
       } return prev;
    }
}
```

2095. Delete the Middle Node of a Linked List

```
            class Solution {
                public ListNode deleteMiddle(ListNode head) {
                    ListNode slow=head;
                    ListNode fast=head;
                    ListNode prev=null;
                     if(head.next==null) return null;
                    while(fast !=null && fast.next !=null){
                        prev=slow;
                        slow=slow.next;
                        fast=fast.next.next;
                    }
                   if(prev !=null)
                    prev.next=slow.next;
                    return head;
            }
            }
```