1. https://leetcode.com/problems/reverse-linked-list/description/?envType=problem-list-v2&envId=linked-list

```
# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, val=0, next=None):
#         self.val = val
#         self.next = next
class Solution:
    def reverseList(self, head: Optional[ListNode]) -> 
Optional[ListNode]:
        if head==None or head.next==None:
            return head
        a=None
        b=head
        c=head.next
        while b:
            b.next=a
            a=b
            b=c
            if c:
                c=c.next
        head=a
        return head
```

2.
```
class node:
    def __init__(self,data):
        self.val=data
        self.next=None
class sll:
    def __init__(self):
        self.head=None
    def insertatbeg(self,data):
        if self.head==None:
            self.head=node(data)
        else:
            new=node(data)
            new.next=self.head
            self.head=new
    def insertatend(self,data):
        if self.head==None:
            self.head=node(data)
        else:
            new=node(data)
            curr=self.head
            while curr.next:
                curr=curr.next
            curr.next=new
    def maximum(self):
        max=0
        curr=self.head
        while curr:
            if curr.val>max:
                max=curr.val
            curr=curr.next
        print(max)
    def insertatpos(self,data,pos):
        new=node(data)
        curr=self.head
        for i in range(pos-2):
            curr=curr.next
        new.next=curr.next
        curr.next=new
    def delatbeg(self):
        self.head=self.head.next
    def delatend(self):
        curr=self.head
        while curr.next.next:
            curr=curr.next
        curr.next=None
    def delofval(self,val):
        curr=self.head
        while curr:
            if curr.next.val==val:
                break
            curr=curr.next
        curr.next=curr.next.next
    def printing(self):
        curr=self.head
        while curr:
            print(curr.val,end="->")
            curr=curr.next

o=sll()
for i in range(6):
    o.insertatend(i)
o.printing()
print()
o.delofval(3)
o.printing()
```

3. https://leetcode.com/problems/palindrome-linked-list/description/?envType=problem-list-v2&envId=linked-list

```
# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, val=0, next=None):
#         self.val = val
#         self.next = next
class Solution:
    def isPalindrome(self, head: Optional[ListNode]) -> bool:
        s=""
        curr=head
        while curr:
            s=s+str(curr.val)
            curr=curr.next
        return s==s[::-1]
```

4. https://leetcode.com/problems/delete-the-middle-node-of-a-linked-list/description/?envType=problem-list-v2&envId=linked-list

```
# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, val=0, next=None):
#         self.val = val
#         self.next = next
class Solution:
    def deleteMiddle(self, head: Optional[ListNode]) ->
Optional[ListNode]:
        if head==None or head.next==None:
            return None
        slow=head
        fast=head
        prev=None
        while fast and fast.next:
            prev=slow
            slow=slow.next
            fast=fast.next.next
        prev.next=prev.next.next
        return head
```

5. https://leetcode.com/problems/remove-duplicates-from-sorted-list/description/?envType=problem-list-v2&envId=linked-list

```
# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, val=0, next=None):
#         self.val = val
#         self.next = next
class Solution:
    def deleteDuplicates(self, head: Optional[ListNode]) ->
Optional[ListNode]:
        curr=head
        while curr and curr.next:
            if curr.val==curr.next.val:
                curr.next=curr.next.next
            else:
                curr=curr.next
        return head
```

6. https://leetcode.com/problems/remove-linked-list-elements/description/?envType=problem-list-v2&envId=linked-list

```
# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, val=0, next=None):
#         self.val = val
#         self.next = next
class Solution:
    def removeElements(self, head: Optional[ListNode], val:
int) -> Optional[ListNode]:
        firstnode=ListNode()
        firstnode.next=head
        curr=firstnode
        while curr and curr.next:
            if curr.next.val==val:
                curr.next=curr.next.next
            else:
                curr=curr.next
        return firstnode.next
```

7. https://leetcode.com/problems/odd-even-linked-list/?envType=problem-list-v2&envId=linked-list

```
# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, val=0, next=None):
#         self.val = val
#         self.next = next
class Solution:
    def oddEvenList(self, head: Optional[ListNode]) ->
Optional[ListNode]:
        if head==None or head.next==None:
            return head
        odd=head
        even=head.next
        temp=even
        while odd.next and odd.next.next:
            odd.next=odd.next.next
            even.next=even.next.next
            odd=odd.next
            even=even.next
        odd.next=temp
        return head
```

8. https://leetcode.com/problems/delete-node-in-a-linked-list/description/?envType=problem-list-v2&envId=linked-list

```
# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, x):
#         self.val = x
#         self.next = None

class Solution:
    def deleteNode(self, node):
        """
        :type node: ListNode
        :rtype: void Do not return anything, modify node in-place instead.
        """
        node.val=node.next.val
        node.next=node.next.next
```

9. https://leetcode.com/problems/remove-duplicates-from-sorted-list-ii/description/?envType=problem-list-v2&envId=linked-list

```
# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, val=0, next=None):
#         self.val = val
#         self.next = next
class Solution:
    def deleteDuplicates(self, head: Optional[ListNode]) -> Optional[ListNode]:
        firstnode=ListNode()
        firstnode.next=head
        prev=firstnode
        curr=head
        while curr:
            while curr.next and curr.val==curr.next.val:
                curr=curr.next
            if prev.next==curr:
```

```python
            prev=prev.next
        else:
            prev.next=curr.next
        curr=curr.next
    return firstnode.next
```