21. Merge Two Sorted Lists

1. Merge Two Sorted Lists

```
# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, val=0, next=None):
#         self.val = val
#         self.next = next
class Solution:
 def mergeTwoLists(self, l1: ListNode, l2: ListNode) -> ListNode:
     if not l1:
         return l2

     if not l2:
         return l1

     if(l1.val < l2.val):

         l1.next=self.mergeTwoLists(l1.next,l2)
         return l1

     else:
         l2.next=self.mergeTwoLists(l1,l2.next)
         return l2;
```

83. Remove Duplicates from Sorted List

Remove Duplicates from Sorted Lists

```
# 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: ListNode) -> ListNode:
        if head:
            prev = head
            curr = head.next

            while curr  != None:
                if prev.val == curr.val :
                    prev.next = curr.next
                    curr = curr.next
                else:
                    prev = prev.next
                    curr = curr.next

            return head
        else:
            return None
```

141. Linked List Cycle

Linked List Cycle

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

class Solution:
    def hasCycle(self, head: ListNode) -> bool:
        slow = fast = head
        while(fast and fast.next):
            slow = slow.next
            fast = fast.next.next
            if(slow == fast):
                return True
        return False
```

206. Reverse Linked List

2. Reverse 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: ListNode) -> ListNode:

     prev=None
     curr=head

     while curr:
         temp=curr.next
         curr.next=prev
         prev=curr
         curr=temp
     return prev
```

234. Palindrome Linked List

Palindrome Linked List

```
## RC ##
        ## APPROACH : similar to reverse linked list ##
        #   1. First make a copy of LL.
        #   2. use fast and slow to find middle of LL #(now slow has second half of LL)
        #   3. now reverse slow LL and compare with first half of original copy.

        ## TIME COMPLEXITY : O(N) ##
        ## SPACE COMPLEXITY : O(1) ##

        #EDGE CASE
        if(not head or not head.next):return True                    #Edge Case

        slow=fast=head                                                #1. SLOW AND FAST(GO
TO MID with SLOW)
        while(fast and fast.next):
            slow=slow.next
            fast = fast.next.next



        #2. REVERSE SLOW LL
        # LOGIC : 3->2->1
        #1. CREATE A REV node with first VALUE.
        #2. ITERATE
        #3. COPY REV TO PREVIOUS # COZ we have to concatinate the prev rev chain to curr
ent
        #4. CONCAT PREV TO CURR NODE # not curr to prev
        #5.
        rev=ListNode(slow.val);
        slow=slow.next
        while(slow):
            prev=rev                                                  # IMP STEP # USE PRE
VIOUS
            curr = ListNode(slow.val);
            curr.next = prev
            rev=curr
            slow=slow.next

        while(rev):
            if(rev.val!=head.val):
                return False
            rev=rev.next
            head=head.next
        return True
```

237. Delete Node in a Linked List

Delete Node in a 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
```

876. Middle of the Linked List

Middle of the 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 middleNode(self, head: ListNode) -> ListNode:
        slow = head
        fast = head

        while fast is not None and fast.next is not None:
            slow = slow.next
            fast = fast.next.next

        return(slow)
```

1290. Convert Binary Number in a Linked List to Integer

Convert Binary Number in a LInked List to Integer

```
# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, val=0, next=None):
#         self.val = val
#         self.next = next
class Solution:
    def getDecimalValue(self, head: ListNode) -> int:
        dec = 0
        while head:
            dec = dec * 2 + head.val
            head = head.next
        return dec
```
