Q1

class Solution:

    def deleteMiddle(self, head: Optional[ListNode]) -> Optional[ListNode]:

        if not head.next:

            return None

        slow, fast = head, head

        mid = None

        while fast and fast.next:

            mid = slow

            slow = slow.next

            fast = fast.next.next

        mid.next = mid.next.next

        return head

Q2

class Solution:

def hasCycle(self, head: Optional[ListNode]) -> bool:

nodes = {}

while (head):

if head.next not in nodes:

nodes[head.next] = 1

else:

return True

head = head.next

return False

Q3

class Solution:

def removeNthFromEnd(self, head: Optional[ListNode], n: int) -> Optional[ListNode]:

dummy = ListNode()

dummy.next = head

pnt1, pnt2 = dummy, head

for \_ in range(n):

pnt2 = pnt2.next

while pnt2:

pnt1, pnt2 = pnt1.next, pnt2.next

pnt1.next = pnt1.next.next

return dummy.next

Q4

class Solution:

def isPalindrome(self, head):

if head is None:

return 0

if head.next is None:

return 1

current = head

list1 = []

while current != None:

list1.append(current.data)

current = current.next

list1

rev = list1[::-1]

if list1 == rev:

return 1

else:

return 0

Q5

class Solution:

#Function to remove a loop in the linked list.

def removeLoop(self, head):

slow=head

fast=head

l=0

while(slow and fast and fast.next):

slow=slow.next

fast=fast.next.next

if(fast==slow):

l=1

break

if l:

start=head

meet=fast

while(start!=meet):

start=start.next

meet=meet.next

while(meet.next!=start):

meet=meet.next

meet.next=None

return 1

return 0

Q6

class Solution:

def skipMdeleteN(self, head, M, N):

temp = head

while(temp):

for i in range(1,M):

if(temp!=None):

temp = temp.next

if(temp == None):

return

t = temp.next

for i in range(1,N+1):

if(t!=None):

t = t.next

temp.next = t

temp = t

return head

Q7

def mergeList(head1, head2):

if head1 and head2:

first = head1

second = head2

while first!=None and second!=None:

temp1 = first.next

first.next = second

temp2 = second.next

second.next = temp1

first = temp1

second = temp2

head2 = second

return [head1,head2]

Q8

def isCircular(head):

# Code here

count=0

temp1=head

temp=head

while temp:

temp=temp.next

if temp==temp1:

return 1

return 0