In [1]:
```
print("Hai Good morning...")
```

```
Hai Good morning...
```

In [2]:
```
class node:
    def __init__(self,data):
        self.data = data
        self.next = None
```

In [3]:
```
n1 = node(10)
print(n1.data) #10
print(n1.next) #None
```

```
10
None
```

In [4]:
```
n2 = node(20)
print(n2.data) #20
print(n2.next) #None
```

```
20
None
```

In [5]:
```
n1.next = n2
print(n1.data) #10
print(n2.data) #20
print(n1.next) #<__main__.node object at 0x000002D5C4815E50>
print(n2.next) #None
```

```
10
20
<__main__.node object at 0x000002B1F8BE9110>
None
```

In [6]:
```
head = node(111)
print(head.data) #111
print(head.next) #None
```

```
111
None
```

In [7]:
```
head.next = node(222)
print(head.data) #111
print(head.next.data) #222
print(head.next.next) #None
```

```
111
222
None
```

In [8]:
```
#display function by using iteration O(n)
def printlist(head):
    temp = head
    if temp==None:
        print("LIST IS EMPTY")
        return
    while temp!=None:
        print(temp.data, end=" => ")
        temp = temp.next
    print("None")
```

In [9]:
```
printlist(head)
```

```
111 => 222 => None
```

In [10]:
```
head.next.next = node(333)
head.next.next.next = node(444)
printlist(head)
```

```
111 => 222 => 333 => 444 => None
```

In [11]:
```
#display function by using recursion O(n) fisr to last
def displayrecursion(temp):
    if temp==None:
        print("None")
        return
    print(temp.data,end=" => ")
    displayrecursion(temp.next)

displayrecursion(head)
```

```
111 => 222 => 333 => 444 => None
```

In [12]:
```
#display function by using recursion O(n) last to first
def displayrecursion(temp):
    if temp==None:
        print("None")
        return
    displayrecursion(temp.next)
    print(temp.data,end=" => ") #non tail recursion

displayrecursion(head)
```

```
None
444 => 333 => 222 => 111 =>
```

In [13]:
```
#insert at first location O(1)
def insertatfirst(head,data):
    temp = node(data)
    temp.next = head
    return temp

printlist(head)
head = insertatfirst(head,999)
printlist(head)
```

```
111 => 222 => 333 => 444 => None
999 => 111 => 222 => 333 => 444 => None
```

In [14]:
```
# insert at last O(n)
def insertatlast(head,data):
    newnode = node(data)
    if head==None:
        head = newnode
        return head
    currnode = head
    while currnode.next!=None:
        currnode = currnode.next
    currnode.next = newnode
    return head
printlist(head)
```

```
head = insertatlast(head,888)
printlist(head)
```

```
999 => 111 => 222 => 333 => 444 => None
999 => 111 => 222 => 333 => 444 => 888 => None
```

In [15]:
```
#insert at specified location: O(n)
def insertatlocation(head,location,data):
    newnode = node(data)
    if location==0:
        newnode.next = head
        return newnode
    currnode = head
    i=0
    while currnode.next!=None and i<location-1:
        currnode = currnode.next
        i=i+1
    newnode.next = currnode.next
    currnode.next = newnode
    return head
```

In [16]:
```
head = None
head = insertatlast(head,999)
head = insertatlast(head,111)
head = insertatlast(head,222)
head = insertatlast(head,333)
head = insertatlast(head,444)
head = insertatlast(head,888)
printlist(head)
```

```
999 => 111 => 222 => 333 => 444 => 888 => None
```

In [17]:
```
printlist(head)
head = insertatlocation(head,3,777)
printlist(head)
```

```
999 => 111 => 222 => 333 => 444 => 888 => None
999 => 111 => 222 => 777 => 333 => 444 => 888 => None
```

In [18]:
```
#counting number of nodes O(n)
def count(head):
    c=0
    curr = head
    while curr!=None:
        c=c+1
        curr = curr.next
    return c
```

In [19]:
```
printlist(head)
print(count(head))
```

```
999 => 111 => 222 => 777 => 333 => 444 => 888 => None
7
```

In [20]:
```
#middle element in list O(n)
def middle(head):
    c=0
    n=count(head)
    curr = head
    while c<n//2:
```

```
        c=c+1
        curr = curr.next
    return curr.data
```

In [21]:
```
printlist(head)
print(middle(head))
```

```
999 => 111 => 222 => 777 => 333 => 444 => 888 => None
777
```

In [22]:
```
# search operation case 1: O(n)
def search1(head,data):
    curr = head
    while curr!=None:
        if curr.data==data:
            return True
        curr=curr.next
    return False
```

In [23]:
```
printlist(head)
print(search1(head,333))
print(search1(head,555))
```

```
999 => 111 => 222 => 777 => 333 => 444 => 888 => None
True
False
```

In [24]:
```
# search operation case 2: O(n)
def search2(temp,data):
    if temp==None:
        return False
    if temp.data==data:
        return True
    return search2(temp.next,data)
```

In [25]:
```
printlist(head)
print(search2(head,444))
print(search2(head,666))
```

```
999 => 111 => 222 => 777 => 333 => 444 => 888 => None
True
False
```

In [26]:
```
#search operation case 3: O(n)
def search3(head,data):
    i=0
    curr = head
    while curr!=None:
        if curr.data==data:
            return i
        curr=curr.next
        i=i+1
    return -1
```

In [27]:
```
printlist(head)
print(search3(head,333))
print(search3(head,555))
print(search3(head,888))
```

```
print(search3(head,999))
print(search3(head,666))
```

```
999 => 111 => 222 => 777 => 333 => 444 => 888 => None
4
-1
6
0
-1
```

In [28]:
```
# search operation case 4: O(n)
def search4(temp,data,index):
    if temp==None:
        return -1
    if temp.data==data:
        return index+1
    return search4(temp.next,data,index+1)
```

In [29]:
```
printlist(head)
print(search4(head,333,-1))
print(search4(head,555,-1))
print(search4(head,888,-1))
print(search4(head,999,-1))
print(search4(head,666,-1))
```

```
999 => 111 => 222 => 777 => 333 => 444 => 888 => None
4
-1
6
0
-1
```

In [30]:
```
head = None
printlist(head)
```

```
LIST IS EMPTY
```

In [31]:
```
#sorted insertion in asc oder O(n)
def sortedinsertasc(head,data):
    newnode = node(data)
    if head==None:
        return newnode
    elif data < head.data:
        newnode.next = head
        return newnode
    else:
        curr = head
        while curr.next!=None and curr.next.data < data:
            curr = curr.next
        newnode.next = curr.next
        curr.next = newnode
        return head
```

In [32]:
```
head = None
head = sortedinsertasc(head,5)
head = sortedinsertasc(head,2)
head = sortedinsertasc(head,7)
head = sortedinsertasc(head,6)
```

```
head = sortedinsertasc(head,4)
printlist(head)
```

```
2 => 4 => 5 => 6 => 7 => None
```

In [33]:
```
#sorted insertion in desc oder O(n)
def sortedinsertdesc(head,data):
    newnode = node(data)
    if head==None:
        return newnode
    elif data > head.data:
        newnode.next = head
        return newnode
    else:
        curr = head
        while curr.next!=None and curr.next.data > data:
            curr = curr.next
        newnode.next = curr.next
        curr.next = newnode
        return head
```

In [34]:
```
head = None
head = sortedinsertdesc(head,5)
head = sortedinsertdesc(head,2)
head = sortedinsertdesc(head,7)
head = sortedinsertdesc(head,6)
head = sortedinsertdesc(head,4)
printlist(head)
```

```
7 => 6 => 5 => 4 => 2 => None
```

In [35]:
```
head = None
head = insertatlast(head,10)
head = insertatlast(head,60)
head = insertatlast(head,80)
head = insertatlast(head,20)
head = insertatlast(head,90)
head = insertatlast(head,40)
head = insertatlast(head,50)
printlist(head)
```

```
10 => 60 => 80 => 20 => 90 => 40 => 50 => None
```

In [36]:
```
#delete at first O(1)
def deleteatfirst(head):
    if head==None:
        return None
    return head.next
```

In [37]:
```
head = None
head = insertatlast(head,10)
head = insertatlast(head,60)
head = insertatlast(head,80)
head = insertatlast(head,20)
head = insertatlast(head,90)
head = insertatlast(head,40)
head = insertatlast(head,50)
printlist(head)
```

```
head = deleteatfirst(head)
printlist(head)
```

```
10 => 60 => 80 => 20 => 90 => 40 => 50 => None
60 => 80 => 20 => 90 => 40 => 50 => None
```

In [38]:
```
#delete at last O(n)
def deleteatlast(head):
    if head == None:
        return None
    if head.next==None:
        return None
    curr = head
    while curr.next.next!=None:
        curr = curr.next
    curr.next = None
    return head
```

In [39]:
```
head = None
head = insertatlast(head,10)
head = insertatlast(head,60)
head = insertatlast(head,80)
head = insertatlast(head,20)
head = insertatlast(head,90)
head = insertatlast(head,40)
head = insertatlast(head,50)
printlist(head)
head = deleteatlast(head)
printlist(head)
```

```
10 => 60 => 80 => 20 => 90 => 40 => 50 => None
10 => 60 => 80 => 20 => 90 => 40 => None
```

In [40]:
```
#delete at location O(n)
def deleteatlocation(head,location):
    if head==None:
        return None
    if location==0:
        return head.next
    i=0
    curr = head
    while i<location-1 and curr!=None:
        curr = curr.next
        i=i+1
    if curr.next==None:
        curr.next=None
    else:
        curr.next = curr.next.next
    return head
```

In [41]:
```
head = None
head = insertatlast(head,10)
head = insertatlast(head,60)
head = insertatlast(head,80)
head = insertatlast(head,20)
head = insertatlast(head,90)
head = insertatlast(head,40)
head = insertatlast(head,50)
printlist(head)
```

```
head = deleteatlocation(head,7)
printlist(head)
```

```
10 => 60 => 80 => 20 => 90 => 40 => 50 => None
10 => 60 => 80 => 20 => 90 => 40 => 50 => None
```

In [42]:
```
#delete element O(n)
def deleteelement(head,data):
    if head==None:
        return None
    if head.data==data:
        return head.next
    curr = head
    while curr!=None:
        if curr.data == data:
            break
        prev = curr
        curr = curr.next
    if curr!=None:
        prev.next = curr.next
    return head
```

In [43]:
```
head = None
head = insertatlast(head,10)
head = insertatlast(head,60)
head = insertatlast(head,80)
head = insertatlast(head,20)
head = insertatlast(head,90)
head = insertatlast(head,40)
head = insertatlast(head,50)
printlist(head)
head = deleteelement(head,90)
printlist(head)
```

```
10 => 60 => 80 => 20 => 90 => 40 => 50 => None
10 => 60 => 80 => 20 => 40 => 50 => None
```

In [44]:
```
head = None
head = insertatlast(head,10)
head = insertatlast(head,60)
head = insertatlast(head,40)
head = insertatlast(head,20)
head = insertatlast(head,40)
head = insertatlast(head,50)
head = insertatlast(head,40)
printlist(head)
head = deleteelement(head,40)
printlist(head)
```

```
10 => 60 => 40 => 20 => 40 => 50 => 40 => None
10 => 60 => 20 => 40 => 50 => 40 => None
```

In [45]:
```
#delete elements O(n)
def deleteelements(head,data):
    if head==None:
        return None
    if head.data==data:
        head = head.next
    else:
        curr=head
```

```
        while curr.next!=None:
            if curr.next.data==data:
                curr.next = curr.next.next
            if curr.next!=None:
                curr = curr.next
    return head
```

In [46]:
```
head = None
head = insertatlast(head,10)
head = insertatlast(head,60)
head = insertatlast(head,40)
head = insertatlast(head,20)
head = insertatlast(head,40)
head = insertatlast(head,50)
head = insertatlast(head,40)
printlist(head)
head = deleteelements(head,40)
printlist(head)
```

```
10 => 60 => 40 => 20 => 40 => 50 => 40 => None
10 => 60 => 20 => 50 => None
```

In [47]:
```
head = None
head = sortedinsertasc(head,10)
head = sortedinsertasc(head,20)
head = sortedinsertasc(head,30)
head = sortedinsertasc(head,20)
head = sortedinsertasc(head,40)
head = sortedinsertasc(head,20)
head = sortedinsertasc(head,50)
head = sortedinsertasc(head,20)
printlist(head)
```

```
10 => 20 => 20 => 20 => 20 => 30 => 40 => 50 => None
```

In [48]:
```
#remove duplicates from sorted list O(n)
def removeduplicates(head):
    curr = head
    while curr!=None and curr.next!=None:
        if curr.data == curr.next.data:
            curr.next = curr.next.next
        else:
            curr = curr.next
    return head
```

In [49]:
```
head = None
head = sortedinsertasc(head,10)
head = sortedinsertasc(head,20)
head = sortedinsertasc(head,30)
head = sortedinsertasc(head,20)
head = sortedinsertasc(head,40)
head = sortedinsertasc(head,20)
head = sortedinsertasc(head,50)
head = sortedinsertasc(head,20)
printlist(head)
head = removeduplicates(head)
printlist(head)
```

```
10 => 20 => 20 => 20 => 20 => 30 => 40 => 50 => None
10 => 20 => 30 => 40 => 50 => None
```

In [50]:
```
#copy of the list O(n)
def copylist(head):
    currnode = head
    if currnode == None:
        return Node
    headnode = node(currnode.data)
    tailnode = headnode
    currnode = currnode.next
    while currnode!=None:
        newnode = node(currnode.data)
        tailnode.next = newnode
        tailnode = newnode
        currnode = currnode.next
    return headnode
```

In [51]:
```
head1 = None
head1 = insertatlast(head1,10)
head1 = insertatlast(head1,20)
head1 = insertatlast(head1,30)
printlist(head1) #10=>20=>30=>None
head2 = copylist(head1)
printlist(head2) #10=>20=>30=>None
```

```
10 => 20 => 30 => None
10 => 20 => 30 => None
```

In [52]:
```
head1 = None
head1 = insertatlast(head1,10)
head1 = insertatlast(head1,20)
head1 = insertatlast(head1,30)
printlist(head1) #10=>20=>30=>None
head2 = copylist(head1)
printlist(head2) #10=>20=>30=>None
head1 = insertatlast(head1,99)
head2 = insertatlast(head2,88)
printlist(head1)
printlist(head2)
```

```
10 => 20 => 30 => None
10 => 20 => 30 => None
10 => 20 => 30 => 99 => None
10 => 20 => 30 => 88 => None
```

In [53]:
```
#reverse the linked list O(n)
def reverse(head):
    curr = head
    prev = None
    while curr!=None:
        temp = curr.next
        curr.next = prev
        prev = curr
        curr = temp
    return prev
```

In [55]:
```
head = None
head = insertatlast(head,111)
head = insertatlast(head,222)
head = insertatlast(head,333)
printlist(head) #111=>222=>333=>None
```

```
head = reverse(head)
printlist(head) #333=>222=>111=>None
```

```
111 => 222 => 333 => None
333 => 222 => 111 => None
```

In [56]:
```
#comparing two list objects O(n)
def equals(temp1,temp2):
    while temp1!=None and temp2!=None:
        if temp1.data!=temp2.data:
            return False
        temp1 = temp1.next
        temp2 = temp2.next
    return True
```

In [58]:
```
head1 = None
head2 = None
head3 = None
head1 = insertatlast(head1,111)
head1 = insertatlast(head1,222)
head1 = insertatlast(head1,333)

head2 = insertatlast(head2,111)
head2 = insertatlast(head2,222)
head2 = insertatlast(head2,333)

head3 = insertatlast(head3,111)
head3 = insertatlast(head3,222)
head3 = insertatlast(head3,444)

printlist(head1)
printlist(head2)
printlist(head3)

print(equals(head1,head2))
print(equals(head1,head3))
```

```
111 => 222 => 333 => None
111 => 222 => 333 => None
111 => 222 => 444 => None
True
False
```

In [59]:
```
head = None
head = insertatlast(head,111)
head = insertatlast(head,222)
head = insertatlast(head,333)
head = insertatlast(head,222)
head = insertatlast(head,111)
printlist(head)
```

```
111 => 222 => 333 => 222 => 111 => None
```

In [67]:
```
#paliandrome or not
head = None
head = insertatlast(head,111)
head = insertatlast(head,222)
head = insertatlast(head,333)
head = insertatlast(head,222)
head = insertatlast(head,111)
```

```
printlist(head)
thead = copylist(head)
thead = reverse(thead)
printlist(thead)
print(equals(head,thead))
```

```
111 => 222 => 333 => 222 => 111 => None
111 => 222 => 333 => 222 => 111 => None
True
```

In [66]:
```
#paliandrome or not
head = None
head = insertatlast(head,111)
head = insertatlast(head,222)
head = insertatlast(head,333)
head = insertatlast(head,444)
head = insertatlast(head,555)
printlist(head)
thead = copylist(head)
thead = reverse(thead)
printlist(thead)
print(equals(head,thead))
```

```
111 => 222 => 333 => 444 => 555 => None
555 => 444 => 333 => 222 => 111 => None
False
```

In [68]:
```
#nth node from begin O(n)
def nthnodefrombegin(head,n):
    i=1
    curr = head
    if curr == None:
        return -1
    if n<=0 or n>count(head):
        return -1
    while curr!=None and i<n:
        curr = curr.next
        i=i+1
    return curr.data
```

In [69]:
```
#nth node from end O(n)
def nthnodefromend(head,n):
    n=count(head)-(n-1)
    i=1
    curr = head
    if curr == None:
        return -1
    if n<=0 or n>count(head):
        return -1
    while curr!=None and i<n:
        curr = curr.next
        i=i+1
    return curr.data
```

In [73]:
```
head = None
head = insertatlast(head,11)
head = insertatlast(head,12)
head = insertatlast(head,13)
head = insertatlast(head,14)
head = insertatlast(head,15)
```

```
head = insertatlast(head,16)
printlist(head)
print(nthnodefrombegin(head,1))
print(nthnodefrombegin(head,2))
print(nthnodefrombegin(head,3))
print(nthnodefrombegin(head,9))
print(nthnodefrombegin(head,-4))
print(nthnodefromend(head,1))
print(nthnodefromend(head,2))
print(nthnodefromend(head,3))
print(nthnodefromend(head,9))
print(nthnodefromend(head,-4))
```

```
11 => 12 => 13 => 14 => 15 => 16 => None
11
12
13
-1
-1
16
15
14
-1
-1
```

In [ ]: