In [1]:
```
print("Hello")
```
```
Hello
```

In [2]:
```
a = 10
```

In [3]:
```
print(a)
```
```
10
```

In [4]:
```
# creation of linked list
class node:
    def __init__(self,data):
        self.data = data
        self.next = None
```

In [5]:
```
head = node(10)
print(head.data) #10
```
```
10
```

In [6]:
```
head.next = node(20)
print(head.data) #10
print(head.next.data) #20
```
```
10
20
```

In [7]:
```
head.next.next = node(30)
print(head.data) #10
print(head.next.data) #20
print(head.next.next.data) #30
```
```
10
20
30
```

In [8]:
```
head.next.next = node(30)
print(head.data) #10
print(head.next.data) #20
print(head.next.next.data) #30
print(head.next.next.next) #None
```
```
10
20
30
None
```

In [9]:
```
#display operation
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 [10]:
```
printlist(head)
```

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

In [11]:
```
#displaying by using recursion
def printlistrecursion(temp):
    if temp == None:
        print("None")
        return
    print(temp.data,end=" => ")
    printlistrecursion(temp.next)
```

In [12]:
```
temp = head
printlistrecursion(temp)
```

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

In [13]:
```
head = None
head = node(111)
head.next = node(222)
head.next.next = node(333)
head.next.next.next = node(444)
```

In [14]:
```
temp = head
printlistrecursion(temp)
```

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

In [15]:
```
def printlistrecursion(temp):
    if temp == None:
        print("None")
        return
    printlistrecursion(temp.next)
    print(temp.data,end=" => ")
```

In [16]:
```
temp = head
printlistrecursion(temp)
```

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

In [17]:
```
#insertion of node at first location
def insertatfirst(head,data):
    temp = node(data)
    temp.next = head
    return temp
```

In [18]:
```
head1 = None
head1 = insertatfirst(head1,333)
printlist(head1)
```
```
333 => None
```

In [19]:
```
head2 = node(40)
printlist(head2)
```
```
40 => None
```

In [20]:
```
head = None
head = insertatfirst(head,444)
head = insertatfirst(head,333)
head = insertatfirst(head,222)
head = insertatfirst(head,111)
printlist(head)
```
```
111 => 222 => 333 => 444 => None
```

In [21]:
```
#insert at last
def insertatlast(head,data):
    if head==None:
        return node(data)
    temp = head
    while temp.next != None:
        temp = temp.next
    temp.next = node(data)
    return head
```

In [22]:
```
printlist(head)
head = insertatlast(head, 777)
head = insertatlast(head, 888)
printlist(head)
```
```
111 => 222 => 333 => 444 => None
111 => 222 => 333 => 444 => 777 => 888 => None
```

In [23]:
```
head = None
head = insertatlast(head, "HHH")
head = insertatlast(head, "WWW")
head = insertatlast(head, "ZZZ")
head = insertatfirst(head, "AAA")
printlist(head)
```
```
AAA => HHH => WWW => ZZZ => None
```

In [24]:
```
#insert node at position
def insertatposition(head,position,data):
    temp = node(data)
    if position==1:
        temp.next = head
        return temp
    cur = head
    for i in range(position-2):
        cur = cur.next
        if cur == None:
            return head
    temp.next = cur.next
    cur.next = temp
    return head
```

In [25]:

```
head = None
head = insertatfirst(head,444)
head = insertatfirst(head,333)
head = insertatfirst(head,222)
head = insertatfirst(head,111)
printlist(head)
head = insertatposition(head,6,999)
printlist(head)
```

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

In [26]:

```
#count number of nodes
def count(head):
    c=0
    temp = head
    while temp!=None:
        c=c+1
        temp = temp.next
    return c
```

In [27]:

```
head = None
head = insertatfirst(head,444)
head = insertatfirst(head,333)
head = insertatfirst(head,222)
head = insertatfirst(head,111)
head = insertatlast(head,555)
head = insertatlast(head,666)
printlist(head)
print(count(head))
```

```
111 => 222 => 333 => 444 => 555 => 666 => None
6
```

In [28]:

```
#return middle element data
def middle(head):
    c=1
    temp = head
    n = count(head)
    while c<n//2:
        c=c+1
        temp = temp.next
    return temp.data
```

In [29]:

```
head = None
head = insertatfirst(head,444)
head = insertatfirst(head,333)
head = insertatfirst(head,222)
head = insertatfirst(head,111)
head = insertatlast(head,555)
head = insertatlast(head,666)
head = insertatlast(head,777)
head = insertatlast(head,888)
printlist(head)
print(middle(head))
```

```
111 => 222 => 333 => 444 => 555 => 666 => 777 => 888 => None
444
```

In [30]:
```
#sorted insertion in asc order
def sortedinsert(head,data):
    temp = node(data)
    if head==None:
        return temp
    elif data < head.data:
        temp.next = head
        return temp
    else:
        cur = head
        while cur.next != None and cur.next.data < data:
            cur = cur.next
        temp.next = cur.next
        cur.next = temp
        return head
```

In [31]:
```
head = None
head = sortedinsert(head,30)
head = sortedinsert(head,40)
head = sortedinsert(head,60)
head = sortedinsert(head,25)
head = sortedinsert(head,35)
head = sortedinsert(head,45)
head = sortedinsert(head,50)
head = sortedinsert(head,55)
printlist(head)
```

```
25 => 30 => 35 => 40 => 45 => 50 => 55 => 60 => None
```

In [32]:
```
#sorted insertion in desc order
def sortedinsertdesc(head,data):
    temp = node(data)
    if head==None:
        return temp
    elif data > head.data:
        temp.next = head
        return temp
    else:
        cur = head
        while cur.next != None and cur.next.data > data:
            cur = cur.next
        temp.next = cur.next
        cur.next = temp
        return head
```

In [33]:
```
head = None
head = sortedinsertdesc(head,30)
head = sortedinsertdesc(head,40)
head = sortedinsertdesc(head,60)
head = sortedinsertdesc(head,25)
head = sortedinsertdesc(head,35)
head = sortedinsertdesc(head,45)
head = sortedinsertdesc(head,50)
head = sortedinsertdesc(head,55)
printlist(head)
```

```
60 => 55 => 50 => 45 => 40 => 35 => 30 => 25 => None
```

In [34]:
```
#delete an element from begining
def deleteatfirst(head):
    if head==None:
        return None
    else:
        return head.next
```

In [35]:
```
head = None
head = sortedinsert(head,30)
head = sortedinsert(head,40)
head = sortedinsert(head,60)
head = sortedinsert(head,25)
head = sortedinsert(head,35)
head = sortedinsert(head,45)
head = sortedinsert(head,50)
head = sortedinsert(head,55)
printlist(head)
head = deleteatfirst(head)
printlist(head)
```

```
25 => 30 => 35 => 40 => 45 => 50 => 55 => 60 => None
30 => 35 => 40 => 45 => 50 => 55 => 60 => None
```

In [36]:
```
#delete an element located at last
def deleteatlast(head):
    if head==None:
        return None
    if head.next==None:
        return None
    temp = head
    while temp.next.next!=None:
        temp = temp.next
    temp.next = None
    return head
```

In [37]:
```
head = None
head = sortedinsert(head,30)
head = sortedinsert(head,40)
head = sortedinsert(head,60)
head = sortedinsert(head,25)
head = sortedinsert(head,35)
head = sortedinsert(head,45)
head = sortedinsert(head,50)
head = sortedinsert(head,55)
printlist(head)
head = deleteatlast(head)
printlist(head)
```

```
25 => 30 => 35 => 40 => 45 => 50 => 55 => 60 => None
25 => 30 => 35 => 40 => 45 => 50 => 55 => None
```

In [38]:
```
#delete at location
def deleteatlocation(head,location):
    temp = head
    if temp==None:
        return None
    if location==0:
        head = head.next
        return head
    i=0
    while i<location-1 and temp!=None:
        temp = temp.next
        i=i+1
    temp.next = temp.next.next
    return head
```

In [39]:
```
head = None
head = sortedinsert(head,30)
head = sortedinsert(head,40)
head = sortedinsert(head,60)
head = sortedinsert(head,25)
head = sortedinsert(head,35)
head = sortedinsert(head,45)
head = sortedinsert(head,50)
head = sortedinsert(head,55)
printlist(head)
head = deleteatlocation(head,7)
printlist(head)
```

```
25 => 30 => 35 => 40 => 45 => 50 => 55 => 60 => None
25 => 30 => 35 => 40 => 45 => 50 => 55 => None
```

In [40]:
```
#delete element
def deleteelement(head,data):
    if head==None:
        return None
    if head.data == data:
        head = head.next
        return head
    temp = head
    while temp!=None:
        if temp.data == data:
            break
        prev = temp
        temp = temp.next
    if temp!=None:
        prev.next = temp.next
    return head
```

In [41]:
```
head = None
head = sortedinsert(head,30)
head = sortedinsert(head,40)
head = sortedinsert(head,60)
head = sortedinsert(head,35)
head = sortedinsert(head,35)
head = sortedinsert(head,45)
head = sortedinsert(head,30)
head = sortedinsert(head,35)
printlist(head)
head = deleteelement(head,35)
printlist(head)
```

```
30 => 30 => 35 => 35 => 35 => 40 => 45 => 60 => None
30 => 30 => 35 => 35 => 40 => 45 => 60 => None
```

In [42]:
```
#delete elements
def deleteelements(head,data):
    if head==None:
        return None
    if head.data == data:
        head = head.next
        return head
    temp = head
    while temp.next!=None:
        if temp.next.data == data:
            temp.next = temp.next.next
        temp = temp.next
    return head
```

In [43]:

```
head = None
head = insertatlast(head,111)
head = insertatlast(head,222)
head = insertatlast(head,333)
head = insertatlast(head,444)
head = insertatlast(head,555)
head = insertatlast(head,666)
head = insertatlast(head,333)
head = insertatlast(head,888)
printlist(head)
head = deleteelements(head,333)
printlist(head)
```

```
111 => 222 => 333 => 444 => 555 => 666 => 333 => 888 => None
111 => 222 => 444 => 555 => 666 => 888 => None
```

In [44]:

```
#searching for an element
def search1(head,data):
    temp = head
    while temp!=None:
        if temp.data == data:
            return True
        temp = temp.next
    return False
```

In [45]:

```
head = None
head = insertatlast(head,111)
head = insertatlast(head,222)
head = insertatlast(head,333)
head = insertatlast(head,444)
head = insertatlast(head,555)
head = insertatlast(head,666)
head = insertatlast(head,777)
head = insertatlast(head,888)
printlist(head)
print(search1(head,444))
print(search1(head,999))
```

```
111 => 222 => 333 => 444 => 555 => 666 => 777 => 888 => None
True
False
```

In [46]:

```
#search for element and return index vlaue
def search2(head,data):
    temp = head
    i=0
    while temp!=None:
        if temp.data == data:
            return i
        temp = temp.next
        i=i+1
    return -1
```

In [47]:
```
head = None
head = insertatlast(head,111)
head = insertatlast(head,222)
head = insertatlast(head,333)
head = insertatlast(head,444)
head = insertatlast(head,555)
head = insertatlast(head,666)
head = insertatlast(head,777)
head = insertatlast(head,888)
printlist(head)
print(search2(head,444))
print(search2(head,999))
print(search2(head,111))
```

```
111 => 222 => 333 => 444 => 555 => 666 => 777 => 888 => None
3
-1
0
```

In [48]:
```
#search for an element using recursion
def search3(temp,data):
    if temp==None:
        return False
    if temp.data == data:
        return True
    return search3(temp.next,data)
```

In [49]:
```
head = None
head = insertatlast(head,111)
head = insertatlast(head,222)
head = insertatlast(head,333)
head = insertatlast(head,444)
head = insertatlast(head,555)
head = insertatlast(head,666)
head = insertatlast(head,777)
head = insertatlast(head,888)
printlist(head)
print(search3(head,444))
print(search3(head,999))
print(search3(head,111))
```

```
111 => 222 => 333 => 444 => 555 => 666 => 777 => 888 => None
True
False
True
```

In [50]:
```
#search for element and return index value using recursion
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 [51]:
```
head = None
head = insertatlast(head,111)
head = insertatlast(head,222)
head = insertatlast(head,333)
head = insertatlast(head,444)
head = insertatlast(head,555)
head = insertatlast(head,666)
head = insertatlast(head,777)
head = insertatlast(head,888)
printlist(head)
print(search4(head,444,-1))
print(search4(head,999,-1))
print(search4(head,111,-1))
```

```
111 => 222 => 333 => 444 => 555 => 666 => 777 => 888 => None
3
-1
0
```

In [52]:
```
head = None
head = sortedinsert(head,10)
head = sortedinsert(head,40)
head = sortedinsert(head,20)
head = sortedinsert(head,40)
head = sortedinsert(head,30)
head = sortedinsert(head,40)
head = sortedinsert(head,90)
head = sortedinsert(head,40)
head = sortedinsert(head,80)
head = sortedinsert(head,40)
printlist(head)
```

```
10 => 20 => 30 => 40 => 40 => 40 => 40 => 40 => 80 => 90 => None
```

In [53]:
```
#removing duplicate elements from sorted linked list
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 [54]:
```
head = None
head = sortedinsert(head,10)
head = sortedinsert(head,40)
head = sortedinsert(head,20)
head = sortedinsert(head,40)
head = sortedinsert(head,30)
head = sortedinsert(head,40)
head = sortedinsert(head,90)
head = sortedinsert(head,40)
head = sortedinsert(head,80)
head = sortedinsert(head,40)
printlist(head)
head = removeduplicates(head)
printlist(head)
```

```
10 => 20 => 30 => 40 => 40 => 40 => 40 => 40 => 80 => 90 => None
10 => 20 => 30 => 40 => 80 => 90 => None
```

In [55]:
```python
#nth from begin
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 [56]:
```python
head = None
head = sortedinsert(head,10)
head = sortedinsert(head,20)
head = sortedinsert(head,30)
head = sortedinsert(head,40)
head = sortedinsert(head,50)
head = sortedinsert(head,60)
head = sortedinsert(head,70)
head = sortedinsert(head,80)
head = sortedinsert(head,90)
head = sortedinsert(head,100)
printlist(head)
print(nthnodefrombegin(head,3))
print(nthnodefrombegin(head,6))
print(nthnodefrombegin(head,12))
```

```
10 => 20 => 30 => 40 => 50 => 60 => 70 => 80 => 90 => 100 => None
30
60
-1
```

In [57]:
```python
#nth from end
def nthnodefromend(head,n):
    i=1
    curr = head
    n=count(head)-(n-1)
    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 [58]:
```python
head = None
head = sortedinsert(head,10)
head = sortedinsert(head,20)
head = sortedinsert(head,30)
head = sortedinsert(head,40)
head = sortedinsert(head,50)
head = sortedinsert(head,60)
head = sortedinsert(head,70)
head = sortedinsert(head,80)
head = sortedinsert(head,90)
head = sortedinsert(head,100)
printlist(head)
print(nthnodefromend(head,1))
print(nthnodefromend(head,4))
print(nthnodefromend(head,8))
print(nthnodefromend(head,14))
```

```
10 => 20 => 30 => 40 => 50 => 60 => 70 => 80 => 90 => 100 => None
100
70
30
-1
```

In [59]:

```
#reverse of linked list
def reverse(head):
    curr = head
    prev = None
    while curr!=None:
        temp = curr.next
        curr.next = prev
        prev = curr
        curr = temp
    return prev
```

In [60]:

```
head = None
head = insertatlast(head,11)
head = insertatlast(head,22)
head = insertatlast(head,33)
head = insertatlast(head,44)
printlist(head)
head = reverse(head)
printlist(head)
```

```
11 => 22 => 33 => 44 => None
44 => 33 => 22 => 11 => None
```

In [61]:

```
#copy of the list
def copylist(head):
    currNode = head
    if currNode==None:
        return None
    headNode = node(currNode.data)
    tailNode = headNode
    currNode = currNode.next
    while currNode!=None:
        tempNode = node(currNode.data)
        tailNode.next = tempNode
        tailNode = tempNode
        currNode = currNode.next
    return headNode
```

In [66]:

```
list1 = None
list2 = None
list1 = insertatlast(list1,10)
list1 = insertatlast(list1,20)
list1 = insertatlast(list1,30)
list1 = insertatlast(list1,40)
printlist(list1)
printlist(list2)
list2 = copylist(list1)
printlist(list1)
printlist(list2)
```

```
10 => 20 => 30 => 40 => None
List is empty
10 => 20 => 30 => 40 => None
10 => 20 => 30 => 40 => None
```

In [65]:
```
list1 = None
list1 = insertatlast(list1,10)
list1 = insertatlast(list1,20)
list1 = insertatlast(list1,30)
list1 = insertatlast(list1,40)
list2 = copylist(list1)

list1 = insertatlast(list1,50)
list2 = insertatlast(list2,60)
printlist(list1)
printlist(list2)
```

```
10 => 20 => 30 => 40 => 50 => None
10 => 20 => 30 => 40 => 60 => None
```

In [67]:
```
#equals or not
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 [69]:
```
list1 = None
list1 = insertatlast(list1,10)
list1 = insertatlast(list1,20)
list1 = insertatlast(list1,30)
list1 = insertatlast(list1,40)
list2 = copylist(list1)
printlist(list1)
printlist(list2)
print(equals(list1,list2))
```

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

In [72]:
```
list1 = None
list1 = insertatlast(list1,10)
list1 = insertatlast(list1,20)
list1 = insertatlast(list1,30)
list1 = insertatlast(list1,40)
list2 = copylist(list1)
list1 = insertatlast(list1,50)
list2 = insertatlast(list2,60)
printlist(list1)
printlist(list2)
print(equals(list1,list2))
```

```
10 => 20 => 30 => 40 => 50 => None
10 => 20 => 30 => 40 => 60 => None
False
```

In [76]:
```
#pali
list1 = None
list2 = None
list1 = insertatlast(list1,10)
list1 = insertatlast(list1,20)
list1 = insertatlast(list1,30)
list1 = insertatlast(list1,20)
list1 = insertatlast(list1,50)

list2 = copylist(list1)
list2 = reverse(list2)
printlist(list1)
printlist(list2)
print(equals(list1,list2))
```

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

In [77]:
```
#pali
list1 = None
list2 = None
list1 = insertatlast(list1,10)
list1 = insertatlast(list1,20)
list1 = insertatlast(list1,30)
list1 = insertatlast(list1,20)
list1 = insertatlast(list1,10)

list2 = copylist(list1)
list2 = reverse(list2)
printlist(list1)
printlist(list2)
print(equals(list1,list2))
```

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

In [ ]: