DAY4

Coding problems on DSA

Insert Class for LL

- insert at beginning
- deleting by node
- printing

In [57]:
```
class linkedlist:
    class Node:
        def __init__(self,data):
            self.data=data
            self.next=None

    def __init__(self):
        self.head=None

    def insbeg(self,data):
        newnode=linkedlist.Node(data)
        newnode.next=self.head
        self.head=newnode

    def insend(self,data):
        if self.head==None:
            return "Empty List"
        newnode=linkedlist.Node(data)
        curr=self.head
        while curr.next:
            curr=curr.next
        curr.next=newnode

    def trav(self):
        curr=self.head
        while curr:
            print(curr.data , end="-->")
            curr=curr.next
        print("None")

    def delbynode(self, val):
            if self.head is None:
                return "Empty List"
            if self.head.data == val:
                self.head = self.head.next
            curr = self.head
            while curr.next:
                if curr.next.data == val:
                    break
                curr = curr.next

            if curr.next is None:
            curr.next = curr.next.next

    def rev_link(self):
        prev = None
        curr = self.head

        while curr:
            next_node = curr.next
            curr.next = prev
            prev = curr
            curr = next_node
```

```
        self.head = prev
```

In [60]:
```
s=linkedlist()
s.insbeg(80)
s.insbeg(60)
s.insbeg(70)
s.insbeg(10)
s.insend(20)
s.delbynode(80)
s.trav()
s.rev_link()
s.trav()
```

```
10-->70-->60-->20-->None
20-->60-->70-->10-->None
```

In [61]:
```
# class LinkedList:
#     class Node:
#         def __init__(self, data):
#             self.data = data
#             self.next = None

#     def __init__(self):
#         self.head = None

#     def insbeg(self, data):
#         newnode = LinkedList.Node(data)
#         newnode.next = self.head
#         self.head = newnode

#     def trav(self):
#         curr = self.head
#         while curr:
#             print(curr.data, end=" --> ")
#             curr = curr.next
#         print("None")

#     # Method to reverse the linked list
#     def rev_link(self):
#         prev = None
#         curr = self.head

#         while curr:
#             next_node = curr.next  # Save the next node
#             curr.next = prev  # Reverse the current node's pointer
#             prev = curr  # Move prev to the current node
#             curr = next_node  # Move to the next node

#         self.head = prev  # Reset the head to the new first node

# # Example Usage:
# ll = LinkedList()
# ll.insbeg(1)
# ll.insbeg(2)
# ll.insbeg(3)
# ll.insbeg(4)

# print("Original Linked List:")
# ll.trav()  # Output: 4 --> 3 --> 2 --> 1 --> None

# ll.rev_link()  # Reverse the linked list

# print("Reversed Linked List:")
# ll.trav()  # Output: 1 --> 2 --> 3 --> 4 --> None
```

In [65]:
```
def summn(n,m):
    res=0
    for i in range(n,m+1):
        if i%3==0 and i%5==0:
            res=res+i
    return res
```

In [66]:
```
summn(12,60)
```

Out[66]:
```
150
```

In [68]:
```
a=linkedlist()
a.insbeg(3)
a.insbeg(4)
a.insbeg(2)
b=linkedlist()
b.insbeg(4)
b.insbeg(6)
b.insbeg(5)
a.trav()
b.trav()
```

```
2-->4-->3-->None
5-->6-->4-->None
```

In [72]:
```
dummy=linkedlist()
dummy=linkedlist.Node(0)
dummy.trav()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[72], line 3
      1 dummy=linkedlist()
      2 dummy=linkedlist.Node(0)
----> 3 dummy.trav()

AttributeError: 'Node' object has no attribute 'trav'
```

In [78]:
```
nums = [-2,1,-3,4,-1,2,1,-5,4]
```

In [79]:
```
nums
```

Out[79]:
```
[-2, 1, -3, 4, -1, 2, 1, -5, 4]
```

In [83]:
```
def maxsub(arr):
    c=arr[0]
    g=arr[0]
    for i in range(1,len(arr)):
        c=max(arr[i],c+arr[i])
        g=max(c,g)
    return g
maxsub([-2, 1, -3, 4, -1, 2, 1, -5, 4])
```

Out[83]: 6

In [82]:
```
maxsub([-2, 1, -3, 4, -1, 2, 1, -5, 4])
```

Out[82]: 6

Implementing Queue using Stack

In [84]:
```
class MyQueue:
    def __init__(self):
        self.s1 = []
        self.s2 = []

    def push(self, x):
        while self.s1:
            self.s2.append(self.s1.pop())
        self.s1.append(x)
        while self.s2:
            self.s1.append(self.s2.pop())

    def pop(self):
        return self.s1.pop()

    def peek(self):
        return self.s1[-1]

    def empty(self):
        return not self.s1
```

In [ ]: