In [4]:

```
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None
class LinkedList:
    def __init__(self):
        self.head = None
    def print_ll(self):
        if self.head is None:
            print("List is empty")
        else:
            t = self.head
            while t is not None:
                print(t.data , end=" ")
                t = t.next
            print()
    def insertion_at_beg(self, data):
        new_node = Node(data)
        new_node.next = self.head
        self.head = new_node
    def add_at_end(self, data):
        end_node = Node(data)
        if self.head is None:
            self.head = end_node
        else:
            t = self.head
            while t.next is not None:
                t = t.next
            t.next = end_node
    def add_at_after_loc(self, data, x):
        new_node = Node(data)
        t = self.head
        while t is not None:
            if t.data == x:
                break
            else:
                t = t.next
        if t is None:
            print("Value not found")
        else:
            new_node.next = t.next
            t.next = new_node
    def insert_before_loc(self,data,x):
        if self.head is None:
            print("List is empty")
        elif self.head.data == x:
            new_node = Node(data)
            new_node.next = self.head
            self.head = new_node
        else:
            t = self.head
            while (t.next is not None):
                if t.next.data == x:
                    break
                t=t.next
            if t.next is None:
```

```
                print("Node not found")
            else:
                new_node = Node(data)
                new_node.next = t.next
                t.next = new_node
    def reverse_ll(self):
        prev = None
        t = self.head
        while t is not None:
            next = t.next
            t.next = prev
            prev = t
            t = next
        self.head = prev
    def deletion_at_beg(self):
        if self.head == None:
            print("List is empty")
        else:
            self.head = self.head.next
llist = LinkedList()
llist.head = Node(30)
second = Node(20)
third = Node(10)
llist.head.next = second
second.next = third
llist.print_ll()
m = int(input("Enter an element to insert at end"))
llist.add_at_end(m)
llist.insert_before_loc(70,20)
llist.print_ll()
llist.deletion_at_beg()
llist.print_ll()
```

```
30 20 10
Enter an element to insert at end50
30 70 20 10 50
70 20 10 50
```

In [ ]: