Question 1

Implement deletion operation from the end of the linked list and Insertion operation from the beginning of the linked list

In [1]:
```
class Node:
    def __init__(self, data):
        self.item = data
        self.ref = None

class LinkedList:
    def __init__(self):
        self.start_node = None

    def insertBeg(self, data):
        new_node = Node(data)
        new_node.ref = self.start_node
        self.start_node= new_node

    def deleteEnd(self):
        if self.start_node is None:
            print("The list has no element to delete")
            return

        n = self.start_node
        while n.ref.ref is not None:
            n = n.ref
        n.ref = None

    def display(self):
        if self.start_node == None:
            print("No node to display")
            return
        temp = self.start_node
        while temp != None:
            print(temp.item, end = " ")
            temp = temp.ref

li = LinkedList()
li.insertBeg(11)
li.insertBeg(22)
li.insertBeg(33)
li.display()
```

```
33 22 11
```

Question 2

Implement binary search using python language. (Write a function which returns the index of x in given array arr if present, else returns -1)

In [2]:
```
def binary_search(arr, low, high, x):

    if high >= low:

        mid = (high + low) // 2

        if arr[mid] == x:   #if element is present at the middle itself
            return mid

        elif arr[mid] > x:  #if element is smaller than mid, then it can only be present in left subarray
            return binary_search(arr, low, mid - 1, x)

        else:               #else the element can only be present in right subarray
            return binary_search(arr, mid + 1, high, x)

    else:   #element is not present in the array return -1
        return -1

# Test array
arr = [ 12, 23, 54, 100, 140 ]
x = 100

# Function call
result = binary_search(arr, 0, len(arr)-1, x)

if result != -1:
    print("Element is present at index", str(result))
else:
    print("Element is not present: -1")
```

```
Element is present at index 3
```

Question 3

Write a Python program to find the middle of a linked list.

In [3]:
```
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None

class LinkedList:

    def __init__(self):
        self.head = None

    def push(self, new_data):
        new_node = Node(new_data)
        new_node.next = self.head
        self.head = new_node

    def printMiddle(self):    # Function to get the middle of  the linked list
        slow_ptr = self.head
        fast_ptr = self.head

        if self.head is not None:
            while (fast_ptr is not None and fast_ptr.next is not None):
                fast_ptr = fast_ptr.next.next
                slow_ptr = slow_ptr.next
            print("The middle element is: ", slow_ptr.data)

list1 = LinkedList()
list1.push(11)
list1.push(22)
list1.push(33)
list1.push(44)
list1.push(55)
list1.printMiddle()
```

```
The middle element is:  33
```