**ASSIGNMENT- 4**

**Ques:1**

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

**Ans:1**

class Node:  
 def \_\_init\_\_(self, data):  
 self.data = data  
 self.next = None  
  
class LinkedList:  
 def \_\_init\_\_(self):  
 self.head = None

def insertEnd(self, data):  
 new = Node(data)  
 if self.head == None:  
 self.head = new  
 return  
 temp = self.head  
 while temp.next != None:  
 temp = temp.next  
 temp.next = new

# Function to insert at the start of the linked list  
 def insertStart(self, data):  
 new = Node(data)  
 new.next = self.head  
 self.head = new  
   
 # Function to remove the last node of the linked list   
 def deleteLastNode(self):  
 if(self.head != None):  
 if(self.head.next == None):  
 self.head = None  
 else:  
 temp = self.head  
 while(temp.next.next != None):  
 temp = temp.next  
 lastNode = temp.next  
 temp.next = None  
 lastNode = None

def display(self):  
 if self.head == None:  
 print("No node to display")  
 return  
 temp = self.head  
 while temp != None:  
 print(temp.data, end = " ")  
 temp = temp.next  
   
  
li = LinkedList()  
li.insertEnd(10)  
li.insertEnd(20)  
li.insertEnd(30)  
li.insertStart(0)  
li.insertStart(-10)  
li.deleteLastNode()  
li.display()

**Ques: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)

**Ans:2**

def binary\_search(arr,element):

low = 0

high = len(arr)-1

if\_found = False

while( low<=high and not if\_found):

mid = (low + high)//2

if arr[mid] == element :

if\_found = True

else:

if element < arr[mid]:

high = mid - 1

else:

low = mid + 1

if if\_found==True:

return("Element {} found at index".format(element),mid)

else:

return -1

arr = []

for i in range(int(input("How many elements are in list : "))):

arr.append(int(input("Enter elements one by one: ")))

print("The array is : ",arr)

element = int(input("Enter element to search : "))

print(binary\_search(arr,element))