**Assignment 4 | 21st January 2021**

### Q uestion 1

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

**Ans**: #insertion

def insertEnd(self,data):

new = Node(data)

if self.head == None: #check if the LL is empty

self.head = new

return

temp = self.head

while temp.next != None:

#until we reach the last element that is not pointing to none

temp = temp.next #shifting the temp

temp.next = new

#deletion

def deleteBeg(self):

if self.head == None:

print("LL is empty!")

return

self.head = self.head.next #storing the reference of second node in head

**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)

l = [int(i) for i in input('enter values : ').split()]

x = int(input('enter the item to search: ')

for i in range(0,len(l)):

if l[i] = = x:

print("Hurray! Item found at index:",i)

break

else:

print("Item not found")

Result -1

**Q uestion 3**

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

|  |
| --- |
| 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):          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(5)  list1.push(4)  list1.push(2)  list1.push(3)  list1.push(1)  list1.printMiddle() |

**Output:**

The middle element is: 2