Assignment 4

* Question 1

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

Program –

class Node:

def \_\_init\_\_(self, item):

self.item = item

self.next = None

class LinkedList:

def \_\_init\_\_(self):

self.head = None

def insertAtBeginning(self, data):

new\_node = Node(data)

new\_node.next = self.head

self.head = new\_node

def deleteNode(self, position):

if self.head == None:

return

temp\_node = self.head

if position == 0:

self.head = temp\_node.next

temp\_node = None

return

def printList(self):

temp\_node = self.head

while (temp\_node):

print(str(temp\_node.item) + " ", end="")

temp\_node = temp\_node.next

if \_\_name\_\_ == '\_\_main\_\_':

llist = LinkedList()

llist.insertAtBeginning(2)

llist.insertAtBeginning(3)

llist.insertAtBeginning(8)

llist.insertAtBeginning(6)

llist.insertAtBeginning(5)

print('Linked list:')

llist.printList()

print("after inserting at the beginning :")

llist.insertAtBeginning(10)

llist.printList()

print("\nAfter deleting an element:")

llist.deleteNode(0)

llist.printList()

* 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)

def binary\_search(list1, n):

low = 0

high = len(list1) - 1

mid = 0

while low <= high:

mid = (high + low) // 2

if list1[mid] < n:

low = mid + 1

elif list1[mid] > n:

high = mid - 1

else:

return mid

return -1

list1 = [12, 24, 32, 39, 45, 50, 54, 89, 99, 75, 56]

print(list1)

n = int(input("enter number you want to search in given list : "))

result = binary\_search(list1, n)

if result != -1:

print("Element is present at index", str(result))

else:

print("element is not present")

* Question 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()