Assignment 4

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

Ans:

```
class Node:

    # Constructor to initialize the node object
    def __init__(self, data):
        self.data = data
        self.next = None

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

    # Function to insert a new node at the beginning
    def push(self, new_data):
        new_node = Node(new_data)
        new_node.next = self.head
        self.head = new_node
    # Function to insert a new node at the beginning
    def deleteNode(self, key):

        temp = self.head

        # If head node itself holds the key to be deleted
        if (temp is not None):
            if (temp.data == key):
                self.head = temp.next
                temp = None
                return

        while(temp is not None):
            if temp.data == key:
                break
            prev = temp
            temp = temp.next
```

```
        # if key was not present in linked list
        if(temp == None):
            return

        # Unlinking the node from linked list
        prev.next = temp.next

        temp = None

    def printList(self):
        temp = self.head
        while(temp):
            print (" %d" %(temp.data)),
            temp = temp.next


# Driver program
llist = LinkedList()
llist.push(7)
llist.push(1)
llist.push(3)
llist.push(2)

print ("Created Linked List: ")
llist.printList()
llist.deleteNode(1)
print ("\nLinked List after Deletion of 1:")
llist.printList()
```

2.Implement binary search using python language.

Ans:

```
def binary_search(arr, low, high, x):

    if high >= low:

        mid = (high + low) // 2

        if arr[mid] == x:
```

```
            return mid

        elif arr[mid] > x:
            return binary_search(arr, low, mid - 1, x)

        else:
            return binary_search(arr, mid + 1, high, x)

    else:
        return -1

# Testing array
arr = [ 2, 3, 4, 10, 40 ]
x = 10

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 in array")
```

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

```
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)

# Driver code
list1 = LinkedList()
list1.push(5)
list1.push(4)
list1.push(2)
list1.push(3)
list1.push(1)
list1.printMiddle()
```