Assignment 4 | 21st January 2021

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

beginning of the linked list

Answer:-

Insertion operation from beginning of linked list

```
class Node:
    def __init__(self, dataval=None):
        self.dataval = dataval
        self.nextval = None
class SLinkedList:
    def __init__(self):
        self.headval = None
    def listprint(self):
        printval = self.headval
        while printval is not None:
            print (printval.dataval)
            printval = printval.nextval
    def AtBegining(self,newdata):
        NewNode = Node(newdata)
        NewNode.nextval = self.headval
        self.headval = NewNode
```

```
list = SLinkedList()
list.headval = Node("Mon")
e2 = Node("Tue")
e3 = Node("Wed")
list.headval.nextval = e2
e2.nextval = e3
list.AtBegining("Sun")
list.listprint()
```

Deletion operation from the end of linked list

```
class Node:
    def __init__(self, data=None):
        self.data = data
        self.next = None
class singly_linked_list:
    def __init__(self):
        self.tail = None
        self.head = None
        self.count = 0
    def append_item(self, data):
        node = Node(data)
        if self.head:
            self.head.next = node
            self.head = node
        else:
            self.tail = node
            self.head = node
        self.count += 1
    def delete_item(self, data):
        current = self.tail
        prev = self.tail
        while current:
            if current.data == data:
                if current == self.tail:
                    self.tail = current.next
                else:
                    prev.next = current.next
                self.count -= 1
                return
            prev = current
            current = current.next
    def iterate_item(self):
        current_item = self.tail
        while current_item:
            val = current_item.data
```

```
            current_item = current_item.next

            yield val

items = singly_linked_list()

items.append_item('PHP')

items.append_item('Python')

items.append_item('C#')

items.append_item('C++')

items.append_item('Java')

print("\nAfter removing the last item from the list:")

items.delete_item('Java')

for val in items.iterate_item():

    print(val)
```

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)

Answer:-

```
def binary_search(arr, x):

    low = 0

    high = len(arr) - 1

    mid = 0

    while low <= high:

        mid = (high + low) // 2

        if arr[mid] < x:

            low = mid + 1

        elif arr[mid] > x:

            high = mid - 1

        else:

            return mid

    return -1

arr = [ 2, 3, 4, 10, 40 ]

x = 10

result = binary_search(arr, x)

if result != -1:

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

else:

    print("-1")
```

Question 3

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

Answer:-

```
class Node:
    def __init__(self, value):
        self.data = value
        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):
        temp = self.head
        count = 0
        while self.head:
            if (count & 1):
                temp = temp.next
            self.head = self.head.next
            count += 1
        print(temp.data)
llist = LinkedList()
```

```
llist.push(1)
llist.push(20)
llist.push(100)
llist.push(15)
llist.push(35)
llist.printMiddle()
```