Name

**RAUNAK RAJESH SHAH**

Email

[**shahrrs2004@gmail.com**](mailto:shahrrs2004@gmail.com)

Cohort

**Cohort-29 FSN**

TOPIC

**Data Structure and Algorithms**

College

**Walchand Institute of Technology**

Assignment 4:

**Implementation of Linked List using Python.**

# **Code:**

# Assignment 4: Linked List by RAUNAK SHAH using PYTHON

class Node:

    def \_\_init\_\_(self, data, link):

        self.\_data = data

        self.\_next = link

    @property

    def data(self):

        return self.\_data

    @property

    def next(self):

        return self.\_next

    @data.setter

    def data(self, new\_data):

        self.\_data = new\_data

    @next.setter

    def next(self, next\_node):

        self.\_next = next\_node

class LinkedSeq:

    def \_\_init\_\_(self):

        self.current = None

        self.head = None

        self.capacity = 0

    def \_\_init\_\_(self, capacity):

        self.current = None

        self.head = None

        self.capacity = capacity

    def isEmpty(self) -> bool:

        return self.head == None

    def size(self) -> int:

        temp = self.head

        size = 0

        while temp != None:

            size+=1

            temp = temp.next

        return size

    def start(self) -> None:

        if self.isEmpty():

            self.current = None

        else:

            self.current = self.head

    def isCurrent(self) -> bool:

        return self.current != None

    def advance(self) -> None:

        if not self.isCurrent():

            raise ValueError("No Current found in the list.\nUse `start()` to set current to first element.")

        elif self.current.next == None:

            self.current = self.head

        else:

            self.current = self.current.next

    def removeCurrent(self) -> None:

        if not self.isCurrent():

            raise ValueError("No Current found in the list.\nUse `start()` to set current to first element.")

        elif self.current == self.head:

            if self.capacity <= self.size():

                self.capacity -= 1

            self.current = self.head = None

        else:

            if self.capacity <= self.size():

                self.capacity -= 1

            temp = self.head

            while (temp.next != self.current):

                temp = temp.next

            temp.next = self.current.next

            self.current = None

    def addMany(self, \*elements):

        if self.head == None:

            self.head = Node(elements[0],None)

            for element in elements[1:]:

                new\_node = Node(element, None)

                temp = self.head

                while(temp.next != None):

                    temp = temp.next

                temp.next = new\_node

        if self.capacity < self.size():

            self.capacity = self.size()

    def addAfter(self, element) -> None:

        if not self.isCurrent():

            raise ValueError("No Current found in the list.\nUse `start()` to set current to first element.")

        if self.current.next == None:

            new\_node = Node(element, None)

            self.current.next = new\_node

        else:

            new\_node = Node(element, self.current.next)

            self.current.next = new\_node

        if self.capacity < self.size():

            self.capacity+=1

    def addBefore(self, element) -> None:

        if not self.isCurrent():

            raise ValueError("No Current found in the list.\nUse `start()` to set the current to first element")

        if self.current == self.head:

            new\_node = Node(element, self.current)

            self.head = new\_node

        else:

            new\_node = Node(element, self.current)

            temp = self.head

            while(temp.next != self.current):

                temp = temp.next

            temp.next = new\_node

        if self.capacity <= self.size():

            self.capacity+=1

    def getCurrent(self) -> int:

        return self.current.data

    def display(self) -> None:

        temp = self.head

        print("[",end="")

        while temp != None:

            print(temp.data,end=" ")

            temp = temp.next

        print("]")

seq = LinkedSeq(5)

seq.addMany(1,2,3,4,5,6)

seq.display()

print("Capacity: ",seq.capacity)

seq.start()

print("Current after executing start(): ",seq.getCurrent())

seq.advance()

print("Current after executing advance(): ",seq.getCurrent())

seq.addAfter(7)

print("Sequence after adding element after current element: ",end="")

seq.display()

seq.addBefore(8)

print("Sequence after adding element below current element: ",end="")

seq.display()

seq.removeCurrent()

print("Sequence after removing current: ",end="")

seq.display()

# **Output:**

