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 1: Bag of Integers

**Implementation of Sequence Data Structure as per the Specification provided.**

# **Code:**

# Implemented by Raunak Shah using Python Language

class IntSeq:

    def \_\_init\_\_(self) -> None:

        self.items = []

        self.capacity = 0

        self.current = -1

    def \_\_init\_\_(self, initialCapacity) -> None:

        self.items = []

        self.capacity = initialCapacity

        self.current = -1

    def start(self):

        if len(self.items) > 0:

            self.current = 0

    def isCurrent(self) -> bool:

        return self.current > -1

    def advance(self) -> None:

        if not self.isCurrent():

            raise ValueError('No Current integer found in the sequence. If current element is not set then,\nset the element using `start()` method\nchange current using `advance()` method.')

        if(self.current == len(self.items)-1):

            raise IndexError('No elements ahead to change the current.')

        self.current = self.current+1

    def removeCurrent(self) -> None:

        if not self.isCurrent():

            raise ValueError('No Current integer found in the sequence. If current element is not set then,\nset the element using `start()` method\nchange current using `advance()` method.')

        self.items.remove(self.items[self.current])

        self.current = -1

    def addAfter(self, element) -> None:

        if(self.current + 1 == self.capacity or self.current < 0):

            self.items.append(element)

        else:

            temp = [self.items.pop() for i in range(self.current+1,self.capacity)]

            temp.reverse()

            self.items.append(element)

            self.items.extend(temp)

        if (self.capacity < len(self.items)):

            self.capacity = len(self.items)

        else:

            pass

    def addBefore(self, element) -> None:

        if(self.current < 0):

            self.items.append(element)

            self.capacity = self.capacity + 1

            return

        temp = [self.items.pop() for i in range(self.current,self.capacity)]

        temp.reverse()

        self.items.append(element)

        self.items.extend(temp)

        self.current = self.current + 1

        if (self.capacity < len(self.items)):

            self.capacity = len(self.items)

        else:

            pass

    def addMany(self, \*elements) -> None:

        self.items.extend(elements)

        if (self.capacity < len(self.items)):

            self.capacity = len(self.items)

        else:

            pass

    def ensureCapacity(self, minimumCapacity) -> None:

        self.capacity = minimumCapacity

    def getCurrent(self) -> int:

        if not self.isCurrent():

            raise ValueError('No Current integer found in the sequence. If current element is not set then,\nset the element using `start()` method\nchange current using `advance()` method.')

        else:

            return self.items[self.current]

    def trimToSize(self) -> None:

        self.capacity = len(self.items)

seq = IntSeq(2)

# Adding many elements at one time

seq.addMany(10,20,30,40)

# Getting current item before setting it

# print(seq.getCurrent()) # Will raise error

# Setting Current

seq.start()

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

# Changing Current

seq.advance()

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

#items

print(f"The Sequence: {seq.items}\n")

# adding elements after and before the current element

seq.addAfter(50)

seq.addBefore(60)

print("Sequence after adding elements before and after the Current element: ",seq.items)

print("New Capacity: ",seq.capacity)

seq.ensureCapacity(10)

print("New Capacity: ",seq.capacity)

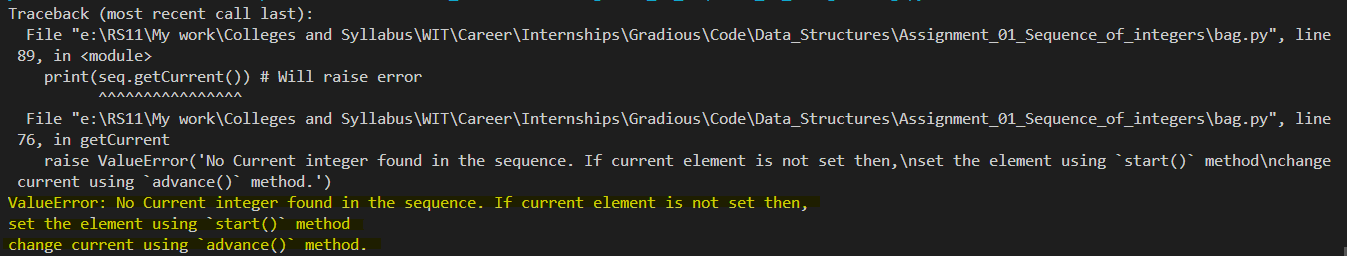
seq.removeCurrent()

print("Sequence after removing current element: ",seq.items)

seq.trimToSize()

print("New Capacity after trimming: ",seq.capacity)

# **Output for throwing error for no current in the sequence:**



# **Output for successful run of program:**

