**SSN College of Engineering Department of Information Technology**

**UIT2201 — Programming and Data Structures 2021 – 2022**

**Exercise — 08**

**1.Provide an implementation of List ADT, using dynamically managed array with the following methods. • Constructor, which takes an argument called ’val’ and creates a new list. If the argument ’val’ is an integer, you should create an array of initial capacity ’val’. If the argument ’val’ is a sequence (such as list, tuple, str, etc.) then your constructor should create an array of double the size of the sequence ’val’ and initialize the newly created list with the elements of the sequence ’val’.**

**• A string method ( *str*() ) to convert this list object into a string.**

**• A length method ( *len*() ) that returns the size of this list object.**

**• A *getitem*() method to return the object stored at the given index.**

**• A *setitem*() method to update the object stored at the given index.**

**• An append method (*append*()), that adds the given object at the end if this list. Note that the underlying array should be dynamically managed and there should not be any capacity limit.**

**• An insert method (*insert*()) to insert the given object at the given index. The size of this list increases by 1, and there should not be any capacity limit. This list is mutated and no new list is created.**

**• A delete method (*delete*()) to delete the object at the given index. This list is mutated and the size reduces by 1. If the size of the list falls below 25% of the current capacity, then resize the underlying array to half of its capacity.**

**• An extend utility (*extend*()) to extend the first list (’self’ if implemented as a method) with the elements of the second list.**

**• *contains*() functionality that returns ‘True’ is the given object is present in the list, ‘False’ other- wise. When implemented as a method, it should be named as *contains* ().**

**• *index*() functionality that returns the index of the first occurrence of the given object in the list. An exception should be raised if the object is not present in the list.**

**• *count*() functionality that the returns the count of occurrences of the given object in the list.**

**2. Implement two different versions of the following list operations. First option is to implement them as basic methods (inside the class) and the second option is to define them as general functions (outside the class). Discuss the merits and demerits of both. Note that the methods have access to the fields of the data structure, while general functions can only use the public methods declared within the class (as given in Question 1).**

**3. Let us perform an empirical analysis to understand the time complexity of the ‘append()’ method. Write a function (takes an integer *n* as an argument) that creates an empty list and append *n* random objects to that list. Your function should record the time taken *T* for these *n* appends, and return the average time *T/n*. Run the experiment for different (very large) values of *n* and note down the average time taken per ‘ap- pend()’ operation. How does this average time increase as *n* increases? Comment on your observation.**

**Program:**

"Importing the modules like ctypes and random"

import ctypes

import random

class array(object):

    """ This class create a dynamic array of capacity"""

    def \_\_init\_\_(self,capacity):

        if isinstance(capacity,int):

            """check wheather the object is int"""

            self.n=0

            self.capacity=capacity

            self.item1=self.make\_array(self.capacity)

        else:

            """if the object is not int data type

            we  have to extend the capacity of array double the size"""

            self.n=len(capacity)

            self.capacity=len(capacity)

            self.item1=self.make\_array(2\*self.capacity)

            for i in range(len(capacity)):

                self.item1[i]=capacity[i]

    def make\_array(self,capacity):

        """Returns a new array with  the given capacity """

        return (capacity\*ctypes.py\_object)()

    def \_\_len\_\_(self):

        """"Return number of elements in the array"""

        return len(self.item1)

    def \_\_str\_\_(self):

        """ return the string of array"""

        string=""

        for i in range(self.n):

            string=string+(str(self.item1[i]))

            if i !=(self.n -1):

                string=string+","

        return "["+string+"]"

    def \_\_getitem\_\_(self,index):

        """return elements at  the index """

        if index<self.n:

            return self.item1[index]

        else:

            raise IndexError ("Index Out of Range")

    def \_\_setitem\_\_(self,index,element):

        """place the element at the given index"""

        if index<self.n:

            self.item1[index]=element

        else:

            raise IndexError ("Index Out of Range")

    def append(self,element):

        """add element at the end of array"""

        if (self.n==self.capacity):

            self.resize(2\*self.capacity)

        self.item1[self.n]=element

        self.n=self.n+1

    def resize(self,capacity):

        """resize the internal array to the given capacity"""

        self.item2=self.make\_array(capacity)

        for i in range(self.n):

            self.item2[i]=self.item1[i]

        self.item1=self.item2

        self.capacity=capacity

    def insert(self,index,element):

        """insert the element at the given index"""

        if not(0<index<self.n) :

            raise IndexError ("Index Out of Range")

        else:

            if (self.n==self.capacity):

                self.resize(2\*self.capacity)

            #move the elements one position towards the right

            for i in range(self.n,index,-1):

                self.item1[i]=self.item1[i-1]

            #inserting the element at the given index

            self.item1[index]=element

            self.n=self.n+1

    def delete(self,index):

        """"delete the element in the array"""

        if not(0<index<self.n) :

            raise IndexError ("Index Out of Range")

        else:

            if (self.n==self.capacity):

                self.resize(2\*self.capacity)

            #move the elements one position towards the right

            for i in range(index,self.n-1,1):

                self.item1[i]=self.item1[i+1]

            self.n=self.n-1

            #shrink the capacity of array

            if self.n<(self.capacity//4):

                self.resize(self.capacity//2)

    def extend(self,elements):

        """add the list to another list"""

        for ele in elements:

            self.append(ele)

    def \_\_contains\_\_(self,element):

        """This function return True if the given element is present in the array else False"""

        for i in range(len(self.item1)):

            if self.item1[i]==element:

                return True

            else:

                return False

    def index(self,element):

        """"This function return the index of the element"""

        for i in range(0,self.n):

            if self.item1[i]==element:

                return "index  of element  {} is {}".format(element,i)

            else :

                return "element not present"

    def count(self,element):

        """This function gives the no of occurrence of the given element in the array"""

        count=0

        for index in range(0,self.n):

            if self.item1[index]==element:

                count=count+1

        return "no of occurrence of  element {} is {}".format(element,count)

#driver code

if \_\_name\_\_ == "\_\_main\_\_":

    obj = array("20")

    for i in range(5):

        obj.append(random.randint(0, 100))

print("Array")

print(obj)

print("\n")

print("Inserting the element 10 at 2")

obj.insert(2,10)

print(obj)

print("\n")

print("Deleting element at 3")

obj.delete(3)

print(obj)

print("\n")

obj.extend([1,2,3,4,5,6,6,7])

print("Extending the element")

print(obj)

print("\n")

print("a=111 present in array")

a=111 in obj

print(a)

print("\n")

print("Return Index of 2")

b=obj.index(2)

print(b)

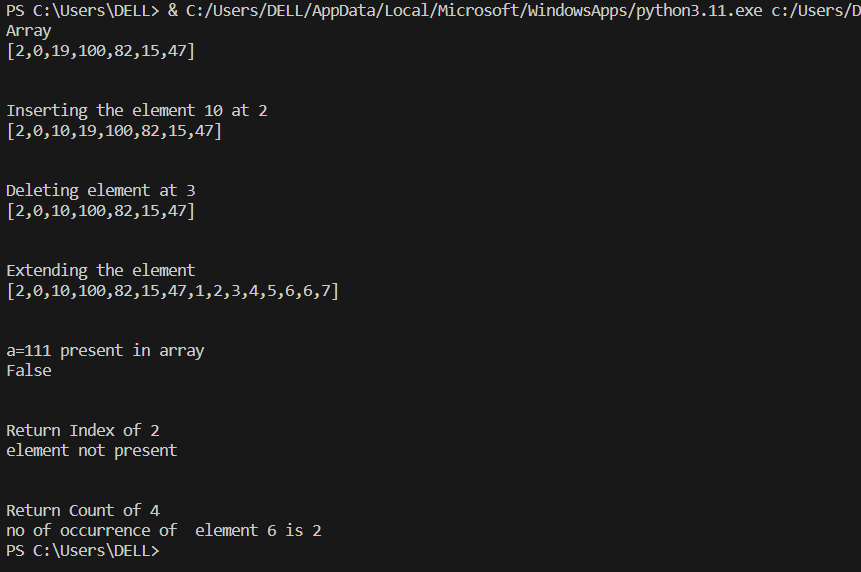
print("\n")

print("Return Count of 4")

c=obj.count(6)

print(c)

**Output:**

****