```
!date
```

```
Mon Feb 28 04:40:10 UTC 2022
```

Please run the above line to refresh the date before your submission.

Recitation 4, CSCI-SHU-210 Data Structure Fall 2021

- For students who have recitation on Wednesday, you should submit your solutions by Friday 11:59pm.
- For students who have recitation on Thursday, you should submit your solutions by Saturday 11:59pm.
- For students who have recitation on Friday, you should submit your solutions by Sunday 11:59pm.

Name: Your Name

NetID: Your NetID

Please submit the following items to the Gradescope:

- Your Colab notebooklink (by clicking the Share button at the top-right corner of the Colab notebook, share to anyone)
- The printout of your run in Colab notebook in pdf format

Note:

No late submission is permitted. All solutions must be from your own work. Total points of the assignment is 100.

Question 1: Implement a Dynamic Array

```
import  ctypes
class  UserDefinedDynamicArray:
        def  __init__(self,I=None):
                self._n=0
                self._capacity=1
                self._A=self._make_array(self._capacity)
                if  I:
                        self.extend(I)

        def  __len__(self):
                return  self._n

        def  append(self,x):
                if  self._n==self._capacity:
```

```
                self._resize(2*self._capacity)
            self._A[self._n]=x
            self._n+=1

    def _resize(self,newsize):
            A=self._make_array(newsize)
            self._capacity=newsize
            for i in range(self._n):
                    A[i]=self._A[i]
            self._A=A

    def _make_array(self,size):
            return (size*ctypes.py_object)()

    def __getitem__(self,i):
            if isinstance(i,slice):
                    A=UserDefinedDynamicArray()
                    for j in range(*i.indices(self._n)): # * operator was used to unp
                            A.append(self._A[j])
                    return A
            if i<0:
                    i=self._n+i
            return self._A[i]

    def __delitem__(self,i):    # Remove by index
            if isinstance(i,slice):
                    for j in reversed(range(*i.indices(self._n))):
                              del self[j]
            else:
                    if i<0:
                            i=self._n+i
                    for j in range(i,self._n-1):
                            self._A[j]=self._A[j+1]
                    self[-1]=None               # Calls __setitem__
                    self._n-=1

                    # Missing some code for Task 8, shrink the size.
                    if (self._n * 4 < self._capacity):
                        self._resize(self._capacity // 2)


    def __str__(self):
            return "[" \
                         +"".join( str(i)+"," for i in self[:-1]) \
                         +(str(self[-1]) if not self.is_empty() else "") \
                         +"]"

    def is_empty(self):
            return self._n == 0

    def __iter__(self):
            # Task 1
            # iterate through the list using yield
            # Your Code
            for i in range(self._n):
                    yield self._A[i]
```

```
                yield self._A[i]

        def __setitem__(self,i,x):
                # Task 2
                # think about how to handle negative index
                # Your code
                if i < 0:
                        i += self._n
                self._A[i] = x

        def extend(self,I):
                # Task 3
                # append all elements of I to the self
                # Your code
                for i in I:
                        self.append(i)

        def reverse(self):
                # Task 4
                # reverse the list
                # your code
                for i in range(len(self) // 2):
                        j = len(self) - 1 - i
                        self._A[i], self._A[j] = self._A[j], self._A[i]

        def __contains__(self,x):
                # Task 5
                # If element x is present in the list return true otherwise false
                # your code
                for each in self:
                        if each == x:
                                return True
                return False

        def index(self,x):
                # Task 5
                # Return the index of first occurrence of element x, if not found in
                # Your code
                for i in range(self.__len__()):
                        if self[i] == x:
                                return i
                return None

        def count(self,x):
                # Task 5
                # return how many times element x is present in the list
                # Your code
                c = 0
                for y in self:
                        if x == y:
                                c += 1
                return c

        def __add__(self,other):
                # Task 6
                # '+' Operator Overloading for UserDefinedDynamicArray Class like myList1+my
```

```
        #  '+' Operator Overloading for UserDefinedDyamicArray Class like myList1+my
        #  Your code
        newList = UserDefinedDynamicArray(self)
        newList.extend(other)
        return newList

    def __mul__(self,times):
        #  Task 6
        #  '*' Operator Overloading for UserDefinedDyamicArray Class like myList1*3
        #  Your code
        newList = UserDefinedDynamicArray()
        for i in range(times):
            newList.extend(self)
        return newList

    __rmul__=__mul__

    def pop(self,i=-1):
        #  Task 7
        #  delete element at position i using del keyword, by default we delete
        #  Your Code
        to_return = self[i]
        del self[i]
        return to_return

    def remove(self,x):          #  Remove by value
        #  Task 7
        #  remove element x from the list, we will delete the first occurrence
        #  at first find out the index of element x, then call __del__(self, i)
        #  Your code
        del self[self.index(x)]

    def max(self):
        #  Task 9
        #  Return the max element in self._A
        #  Your code
        if (self._n == 0):
            return None
        curr = self._A[0]
        for each in self:
            if each > curr:
                curr = each
        return curr

    def min(self):
        #  Task 9
        #  Return the min element in self._A
        #  Your code
        if (self._n == 0):
            return None
        curr = self._A[0]
        for each in self:
            if each < curr:
                curr = each
        return curr
```

```
    def sort(self, order = "asc"):
        # Task 10
        # Sort self._A in ascending order if order == "asc"
        # otherwise sort in descending order if order = 'desc'
        # if order parameter value is wrong, do nothing.
        # Your code
        if order == "asc":
            for j in range(1, self._n):
                key = self._A[j]
                i = j - 1
                while (i > -1) and key < self._A[i]:    # if i == -1 me
                    self._A[i + 1] = self._A[i]    # move the last obje
                    i = i - 1
                self._A[i + 1] = key
        else:
            for j in range(1, self._n):
                key = self._A[j]
                i = j - 1
                while (i > -1) and key > self._A[i]:    # if i == -1 me
                    self._A[i + 1] = self._A[i]    # move the last obje
                    i = i - 1
                self._A[i + 1] = key
```

Task 1: Print the lists

Create two empty list myList1 and myList2, append some elements and print it. You need to implement **__len__** and **__iter__** methods in the UserDefinedDyanmicArray class.

```
myList1 = UserDefinedDynamicArray()
print("myList1: ",myList1)
myList1.append(3)
print("myList1 after appending 3: ",myList1)
myList2=UserDefinedDynamicArray()
for i in range(10):
    myList2.append((i+1)*20)
print("myList2: ",myList2)
```

```
myList1:  []
myList1 after appending 3:  [3]
myList2:  [20,40,60,80,100,120,140,160,180,200]
```

Task2: Delete elements from the myList2 using "del" keyword.

__delitem__ method is already given but you need to write **setitem** method to make it run.

Suppose we want to delete 2nd, third, and fourth elements from myList2 by as follows. This will give you an error as **__setitem__** method needs to be complete

```
print("--------------------Task  2--------------------")
del  myList2[2:5]
print("myList2  after  deleting  index  2,3,4  :  ",myList2)
for  i  in  range(3):
    myList2.append((i+1)*200)
```

```
--------------------Task 2--------------------
myList2 after deleting index 2,3,4 :  [20,40,120,140,160,180,200]
```

Task3: Extending the list using extend function and creating a list from an existing list

Suppose we want to use extend myList1 by adding all the elements in myList2 by calling the extend(self, l) function in the UserDefinedDynamicArray Class

```
myList1.extend(myList2)
print("myList1  after  extending:  ",myList1)
```

```
myList1 after extending:  [3,20,40,120,140,160,180,200,200,400,600]
```

Task4: Reverse a list

```
myList2.reverse()
print("myList2  after  reversing:  ",myList2)
```

```
myList2 after reversing:  [600,400,200,200,180,160,140,120,40,20]
```

Task5: Implement __contains__(self,x), count(x), and index(x)

__contains__ will check whether element x is present in the list. If yes return true, otherwise false

index() will return the index of element x in the list. If x is present multiple times, it will return the first index of x, otherwise it will return None

count() will return how many times element x is present in the list. If the element x is not present, it will return 0.

```
x=140
print("Value of x is: ", x)
print("Whether x is present in the myList1: ",x in myList1) #contains function check
print("x current position in the myList1 is ",myList1.index(x))
#print("Number of times x appears in the myList1 is ",myList1.count(x))
```

```
Value of x is:  140
Whether x is present in the myList1:  True
x current position in the myList1 is  4
```

Task6: Implement __add__(self,other) and __mul__(self,times)

__add__ will implement '+' Operator Overloading for UserDefinedDyamicArray Class, like **myList1+myList2** will return a list containing all the elements of myList1 and then myList2

__mul__ will implement '*' Operator Overloading for UserDefinedDyamicArray Class, like **myList1*3** will return a list having myList1 elements three times.

```
myList3=myList1+myList2
print("myList3 after adding : ",myList3)
myList4 = 2*myList1
print("myList4 after multiplying : ",myList4)
```

```
myList3 after adding :  [3,20,40,120,140,160,180,200,200,400,600,600,400,200,200,180,160,140,
myList4 after multiplying :  [3,20,40,120,140,160,180,200,200,400,600,3,20,40,120,140,160,180
```

Task7: Implement pop(i) function and remove method

By default **pop()** will return the last element from the list and delete that element from the list using del keyword. If i value is specified then we will delete the element at position i and return it to the calling method.

remove(x) will delete the element x from the list. If x is present multiple time, it will delete the first occurrence of x.

```
p=myList2.pop(1)
print("Popped element at position 1 from myList2 ",p)
myList1.remove(140)
print("myList1 after removing: ",myList1)
```

```
Popped element at position 1 from myList2  400
myList1 after removing:  [3,20,40,120,160,180,200,200,400,600]
```

Task8: Modify __delitem__(self,i) function

Current **__delitem__(self, i)** function does not shrink the array capacity.

We want to shrink the array capacity by half if total number of actual elements reduces to one fourth of the capacity.

```
print(myList2, "capacity:", myList2._capacity)
for i in range(7):
    del myList2[0]
print(myList2, "capacity:", myList2._capacity)
```

```
[600,200,200,180,160,140,120,40,20] capacity: 16
[40,20] capacity: 8
```

Task9: Implement max(self); min(self) functions

max(self) function which return maximum element among the elements of self._A.

min(self) function which will return minimum element among the elements of self._A.

```
print("Max of list: ", myList2.max())
print("Min of List: ", myList2.min())
```

```
Max of list: 40
Min of List: 20
```

Task10: Implement sort(self, order='asc')

sort function which will sort the list by default ascending order otherwise descending order if order = 'desc'

```
for i in range(5, 0, -1):
    myList2.append(i)
myList2.sort()
print("After ascending sort: ", myList2)
myList2.sort(order = 'desc')
print("After descending sort: ", myList2)
```

```
After ascending sort: [1,2,3,4,5,20,40]
After descending sort: [40,20,5,4,3,2,1]
```

✓ 0 秒 完成时间：12:40