CSCI-SHU 210 Data Structures

Recitation 4 Array Based Sequences and Dynamic Array

You have a series of tasks in front of you. Complete them! Everyone should code on their own computer, but you are encouraged to talk to others, and seek help from each other and from the TA/LA.

Important:

- **Understand what is a "low-level array"**
  - Also called "static array", "compact array"
  - Fixed capacity, continuous chuck of memory, each cell stores the same type.
  - Supports indexing in O(1) time.
- **Understand what is a "dynamic array"**
  - Supports append( ), pop( ) in O(1) amortized time.
  - Capacity can grow and shrink.
- **Understand what is a "python list"**
  - Each cell can store different type. How?

Question 1 (Implement a Dynamic Array)

Again, what is a Dynamic Array?

Answer: Dynamic array is an array based data structure, that can **shrink/grow its capacity automatically** when it is too full or too empty.

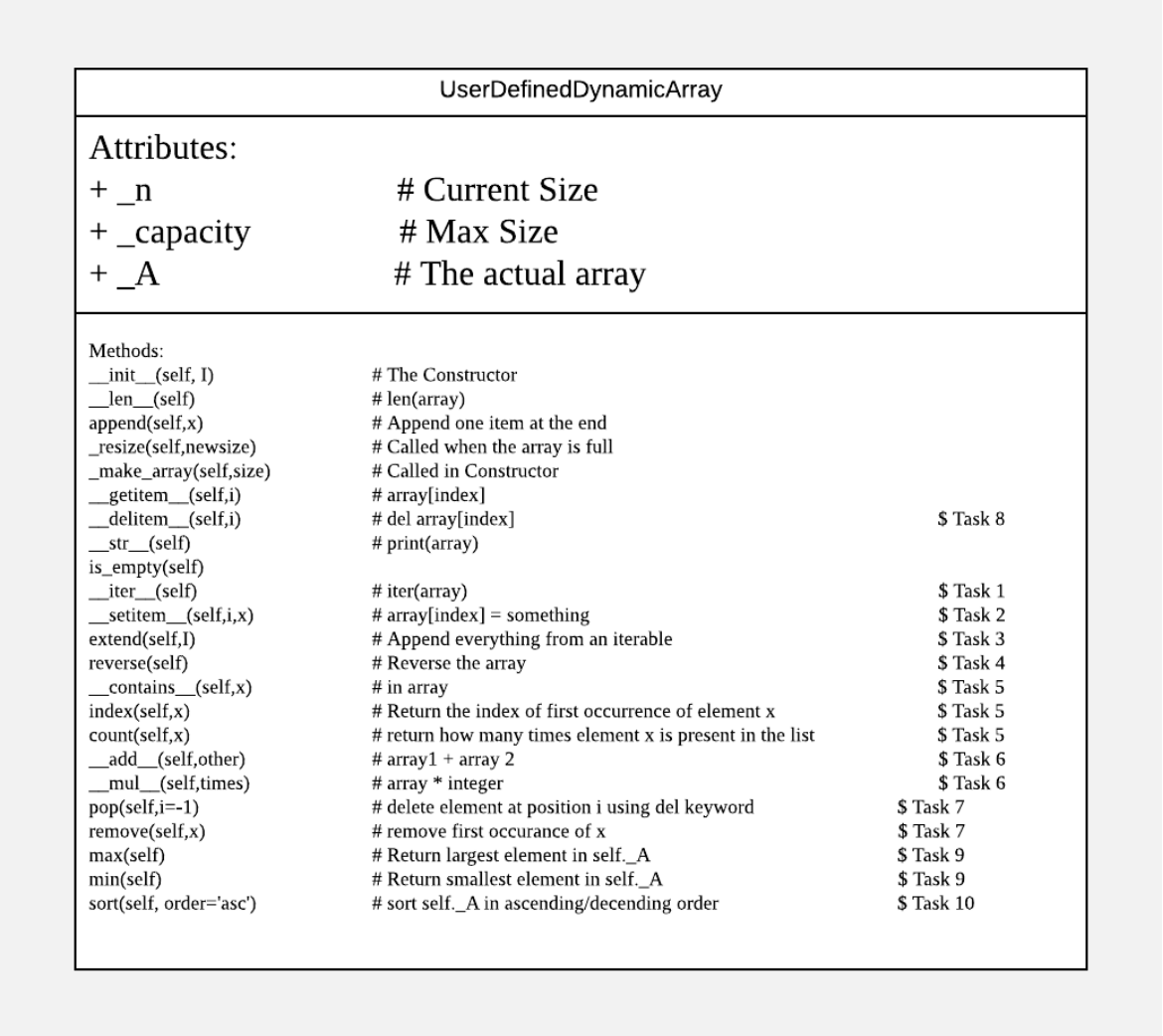

Figure 1: The `UserDefinedDynamicArray` Class UML Diagram.

Your task: Implement `UserDefinedDynamicArray`, so it behaves like our list class. Refer to `DynamicArray.py` for details.

Detailed Tasks breakdown:

Task # 1: Print the lists. To print lists, we have to:
Implement `__iter__` function.

- print function calls `__str__` function, and `__str__` function calls `__iter__` function to get all elements.
- `__iter__` function returns an iterable object by '`yield`' keyword.
- FYI, `__iter__` is called by "`for xxx in list`" code.

```
>>> l = [1,2,3,4]
>>> b = l.__iter__()
>>> b
<list_iterator object at 0x10ade4208>
```

Task # 2: Delete elements from the list.

Implement `__setitem__` function.

- FYI, `__delitem__(self, i)` supports '`del`' keyword.
- FYI, `__setitem__(self, i, x)` supports '`list[index] = value`' operation.
- Question: Why does `__delitem__` need `__setitem__` ?
    - We need to shift elements if the element was deleted in the middle.

```
>>> l = [1,2,3,4]
>>> del l[0:2]
>>> l
[3, 4]
>>> l[1] = 99
>>> l
[3, 99]
```

Task # 3: Extend another iterable.
Implement `extend(self,I)` method.

- Add everything from `I` parameter into `self._A`

```
>>> l = [1,2,3,4]
>>> l2 = [4,5,6]
>>> l.extend(l2)
>>> l
[1, 2, 3, 4, 4, 5, 6]
```

Task # 4: Reverse a list.
Implement `reverse(self) method`.

- Reverse `self._A`.

```
>>> l = [1,2,3,4]
>>> l.reverse()
>>> l
[4, 3, 2, 1]
```

Task # 5: Code three functions.
Implement `__contains__(self, x)`; `index(self, x)`; `count(self, x)` methods.

- `__contains__(self, x)` will check whether element `x` is present in the list. If yes return True, otherwise False. #### `__contains__` supports 'in' operator
- `index(self, x)` 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(self, x)` will return how many times element `x` is present in the list. If the element `x` is not present, it will return 0.

```
>>> l = [1,2,3,4,1]
>>> 1 in l
True
>>> l.index(1)
0
>>> l.count(1)
2
```

Task # 6: Supporting `Array + Array`; `Array * Integer` operations
Implement `__add__(self,other)` and `__mul__(self,times)` methods.

- `__add__` will implement '+' Operator Overloading for `UserDefinedDyamicArray` Class. `myArray1+myArray2` will return a `UserDefinedDyamicArray` containing all the elements of `myArray1` and then `myArray2`
- `__mul__` will implement '*' Operator Overloading for `UserDefinedDyamicArray` Class. `myArray1*3` will return a `UserDefinedDyamicArray` having `myArray1` elements three times.

```
>>> l1 = [1,2,3,4]
>>> l2 = [4,5,6]
>>> l1 + l2
[1, 2, 3, 4, 4, 5, 6]
>>> l1 * 3
[1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4]
```

Task # 7: `pop(i)` and `remove(value)`
Implement `pop(i)` and `remove(value)` methods.

- By default, `pop(i)` 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()` function takes a 'value', and performs element searching to remove this value.
- `__delitem__()` takes an 'index' instead.
- `remove(x)` will delete the element `x` from the list. If `x` is present multiple times, it will delete the first occurrence of `x`.

```
>>> l1 = [1,2,3,4,1]
>>> l1.pop(2) # Pop index 2
3
>>> l1.remove(1)  # Remove first occurrence of value 1
>>> print(l1)
[2, 4, 1]
```

Task # 8: Modify `__delitem__` function so it can shrink the array capacity.

- 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.

```
>>> l1 = [20,40,60,80,100,120,140,160,180,200]
>>> print(l1, "capacity:", l1._capacity)
[20,40,60,80,100,120,140,160,180,200] capacity: 16
>>> for i in range(7):
>>>         del l1[0]
>>> print(l1, "capacity:", l1._capacity)
[160,180,200] capacity: 8
```

Task # 9: Returning `Max/Min` elements of the array.

- `max(self)` function which return maximum element among the elements of `self._A`.
- `min(self)` function which return minimum element among the elements of `self._A`.

```
>>> l1 = [4,7,3,1,9]
>>> l1.max()
9
>>> l1.min()
1
```

Task # 10: Sorting `UserDefinedDynamicArray`.
Implement `sort(self, order = 'asc')` method.

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

```
>>> l1 = [4,7,3,1,9]
>>> l1.sort()
>>> l1
[1, 3, 4, 7, 9]
>>> l1.sort(order = 'desc')
>>> l1
[9, 7, 4, 3, 1]
```