CS256 Lab 5 - And now for something slightly different...

Value: 30 points
Due: 27 September, 2019 @ 23:50
Submission Format: Design document, a modified version of the book's Python script, your Python script, and properly formatted output file submitted to Moodle (four documents in total)

As often as not software engineers modify existing code rather than building completely from scratch. This lab uses a class definition and script from the textbook's website as the basis, your work will be to design, build, and test modifications and additions to that. The class definition is for an expandable array of ints which supports the usual operations. Your job will be to improve some of the existing functionality and add some new features.

Setup

The script dynamic_array.py is on Moodle along with the script da_driver.py that I demonstrated in class. The book's script provides a class DynamicArray which you will improve and extend. There are data files provided that you can test with and use to develop the final output to be submitted.

Tasks

1. Design and write a function that given an input file of integers creates and populates a `DynamicArray` object and returns it to the caller.

2. Design and write a function that given a populated `DynamicArray` object displays the distinct (i.e. unique) integer values from that list. (Hint, since the list has no order you will have to "know" which values you have already displayed.)

3. The `__getitem__` function does not support negative indices, provide a new implementation for this function so that it supports negative indices. (Note: an index such as −3 is equivalent to the traditional index n−3 for a list of length n.)

4. The implementation of the `insert` function is not efficient because when a resize occurs, the resize operation takes time to copy all the elements from an old array to a new array, and then the subsequent loop in the body of insert shifts many of those elements again. Design and write a new function `insertEfficient`, so that, in the case of a resize, the elements are shifted into their final position during that operation, thereby avoiding the subsequent shifting. (Hint: you will need two (non-nested) loops.)

5. The function `remove` removes only the first occurrence of a value from a list. Design and write a new function `removeAll` that removes all occurrences of value from the given list, such that the worst-case running time of the function is O(n) on a list with n elements. (Note: `removeAll` should not rely on repeated calls to `remove`, that would not be O(n).) (Hint: you must be very careful modifying a list at the same time that you are looping through its elements.)

6. Roughly speaking your `main()` will need to process the command line argument, populate a `DynamicArray` with the contents of the data file passed on the command line (using your new `insertEfficient` function), display the unique values found in the array, display the element at location -15, remove all the 7s from the array, and then display the element at location -15 again.

7. From the command line redirect the output from your driver script into a file named **lab-05-final.dat** Your output file should look like so:

   list of unique integers from data-final.dat
   value at arr[-15] before 7s are removed
   value at arr[-15] after 7s are removed

Implementation Notes

1. Design your solution on paper first. Most people would start with the low-level elements (the class definition) and then to the problem of determining and displaying a distinct list. Either way make sure your design covers all of the assignment, you will be turning-in an image of it.
2. The input file name should be a required command line argument (-i).
3. Check to make sure the file exists before trying to open it.
4. We will provide a CSV formatted data files of integers for you to use in testing. One is small, one is large. Use the small one for testing (and/or make your own), use the large one when running the assignment to generate the output file you will turn-in.