CSC-161-500 Lesson 3: Pointers and Dynamic Arrays

Homework Three

Implement the Sequence Class with Dynamic Allocation

Please do Chapter Four, Exercise 2 on page 218. Revise the **sequence1** class of Section 3.2 (and of Homework Two) to use a dynamic array to store its sequence of values. Use the ***sequence2.h*** header file provided by our textbook as your guide to implementing the **sequence2** class. Submit one file for this exercise:

- **sequence2.cxx** - an implementation file for the sequence class, version 2

Please use this file name with the version number. We will later create additional versions of the sequence class to exercise other topics. Your completed class will be tested against a standard test program that creates objects of type **sequence** and exercises all of the required member functions. You will need to test your **sequence** class with your own test program (the interactive test program ***sequence_exam2.cxx*** found in the textbook sample programs is a very complete test), but only submit the implementation file to the Homework Assignment. You will be awarded up to 15 of the 20 points for the assignment based on accurate execution of your **sequence2** class. The remaining 5 points will be awarded subjectively, based on the the quality of your implementation.

One quality item that I want in this lesson, and in subsequent lessons. Document the "invariant" of the class in a comment at the top of the class. The invariant is an explanation of how the internal variables and methods work together. You can be brief but you must be accurate. Beginning with this lesson, I will start counting off points for any class that does not have an invariant.

[If you prefer, you may begin with the sequence1 class from the previous homework solution, which I will make available to all who handed in the assignment. I prefer that you build on your own code, but if you struggled with the previous assignment, you may start "fresh" with the solution. If you did not hand in the previous homework, and do not intend to, you may ask for the solution as the starting point to this assignment. But if that is the case, you should also talk to me about the problems, while we still have time to adjust course enrollments.]

Your **sequence** class for this assignment will differ from the your previous class in the following ways:

- The number of items which may be stored in the sequence should only be limited by the amount of memory available on the heap. When new items are added to a sequence which is at capacity, the size of the data array in which items are stored should be automatically enlarged.
- Because you are dynamically allocating memory within your **sequence** class, **you will need to define a copy constructor, an assignment operator, and a destructor.**
- The constructor should have a default argument which allows the user to set the initial capacity of the sequence.
- There should be a resize function that allows the user to explicitly set the the capacity of the sequence.

CSC-161-500 Lesson 3: Pointers and Dynamic Arrays

Purpose of Exercise

The purpose of this exercise is to ensure that you can write a small class that uses a dynamic array as a private member variable.

Suggestions

1. Start by declaring the new sequence's private member variables in sequence2.h. This should include the dynamic array (which is declared as a pointer to an Item). You will also need two size_t variables to keep track of the number of items in the sequence and the total size of the dynamic array. After you've declared your member variables, write an invariant for the top of sequence2.cxx.
2. As usual, do your work in small pieces. The first version of the sequence may only have a constructor, start, insert, advance, and current. Other member functions can start out as stubs. Use the interactive test program to track down errors in your implementation. If you have an error, *do not start making changes until you have identified the cause of the error.*
3. When a member functions needs to increase the size of the dynamic array, it is a good idea to increase that size by at least 10% (rather than by just one item). The user is likely to keep adding items.