The Assignment:

You will implement and test the sequence class using an array to store the sequence's items.

Purposes:

Ensure that you can write a small class that uses an array as a private member variable.

Familiarize yourself with the sequence container class (which may also be part of future assignments).

Give us a chance to evaluate your programming skills on a small class.

Before Starting:

Read all of Chapter 3.

_____

Files that you must write:

1. `sequence1.h`: The header file for the sequence class. Actually, you don't have to write much of this file. Just start with our version and add your name and other information at the top. Also, decide on appropriate private member variables, an declare these in the sequence class definition at the bottom o the header file If some of your member functions are implemented as inline functions, then you may put those implementations in this file too.
2. `sequence1.cxx`: The implementation file for this first sequence class. You will write all of this file, which will have the implementations of all the sequence's member functions.

Other files that you may find helpful:

1. `sequence_test.cxx:` A simple interactive test program.
2. `sequence_exam.cxx:` A non-interactive test program that will be used to grade the correctness of your sequence class.

The Sequence Class Using a Fixed-Sized Array
Discussion of the Assignment

Many of the features of this class are similar to the Bag class from Section 3.1, so start by throughly reading Section 3.1 and pay attention to new features such as *static constant members* and the use of a *typedef*. The sequence class itself is discussed in Section 3.2 of the class text. Notice how the sequence differs from a Bag (see page 115). The interactive test program `sequence_test.cxx:` is discussed in Section 3.3 of the class text.

Start by declaring the sequence's private member variables in sequence1.h. Then write the invariant of your ADT at the top of sequence1.cxx. The invariant describes precisely how all of your private member variables are used. All of the member functions (except for the constructor) may count on the invariant being true when the member function is activated. And all of the member functions are responsible for ensuring that the invariant is true when the function returns.

As always, do your work in small pieces. For example, my first version of the sequence had only a constructor, start, insert, advance, and current. My other member functions started out as stubs.

Use the interactive test program and the debugger 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.*

The Assignment:

You will implement and test the sequence class using an array to store the sequence's items. A complete definition is here: Sequence with Fixed Sized Array.pdf

Purposes:

- Ensure that you can write a small class that uses an array as a private member variable.
- Familiarize yourself with the sequence container class (which may also be part of future assignments).
- Give us a chance to evaluate your programming skills on a small class.

Before Starting:

Read all of Chapter 3.

Files that you must write:

1. sequence1.h

Download sequence1.h

: The header file for the sequence class. Actually, you don't have to write much of this file. Just start with our version and add your name and other information at the top. Also, decide on appropriate private member variables, an declare these in the sequence class definition at the bottom o the header file If some of your member functions are implemented as inline functions, then you may put those implementations in this file too.

2. sequence1.cxx: The implementation file for this first sequence class. You will write all of this file, which will have the implementations of all the sequence's member functions. NO using copy(). Please create your own for loops to move data around.

Other files that you may find helpful:

1. sequence_test.cxx

: A simple interactive test program.
2. sequence_exam.cxx

: A non-interactive test program that will be used to

grade the correctness of your sequence class.

Submission:

Include required information in the .cpp file comments and in the output (Name and Class). The output should be from running your code with the "...exam.cxx" file.

Copy the output results to a separate file (like HeightOut.txt). Note you will need more than one run of the program with different input values. Copy each of those into one file or submission.

Upload the files to Canvas

Upload as .cpp, .cxx, .docx, .h, .hpp, .rtf or .txt Only!