**A2.**

You will create, implement and test a sequence2 class. The material from Ch1 ~ 4 of the textbook can help you tremendously. You are welcome to use more advance skills than the techniques introduce in the textbook to do the assignment.

This sequence2 class is an extension of the sequence1 class which is introduced in Chapter3 of the text book.

Following is an introduction to some files in this program.

1. sequence2.h, the headers file for the new Sequence class is provided. You can start with the version and add your name and other documentation information at the top. If some of your member functions are implemented as inline functions, then you may put those implementations in this file too. By the way, you might want to compare this header file with the first sequence header file (sequence1.h) in the page 128 of the textbook. The new version no longer has a CAPACITY constant because the items are stored in a dynamic array that grows as needed. But there is a CAPACITY constant, which provides the initial size of the array for a sequence created by the default constructor.
2. sequence2.cpp: The implementation file for the new sequence class. You will write this entire file, which should include all the implementations of the sequence2's member functions.
3. sequence\_test.cpp: This is an interactive test program that we will use to test the program.
4. sequence\_exam2.cpp: A non-interactive test program that will be used to grade the correctness of your new sequence class.

Your sequence class (sequence2) for this assignment will differ from the previous sequence (sequence1) 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 allocation 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 capacity of the sequence.

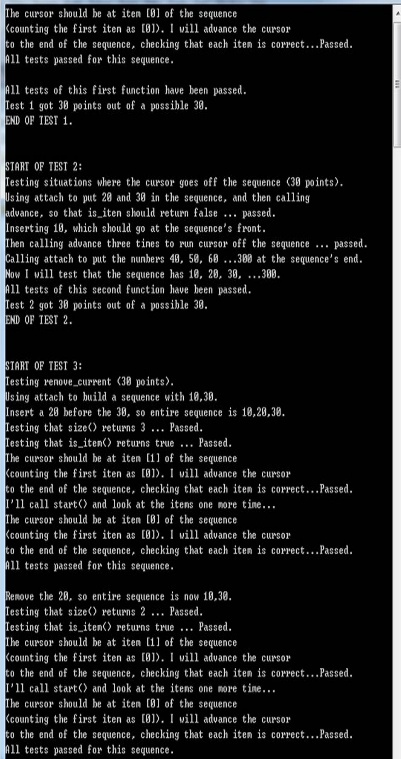
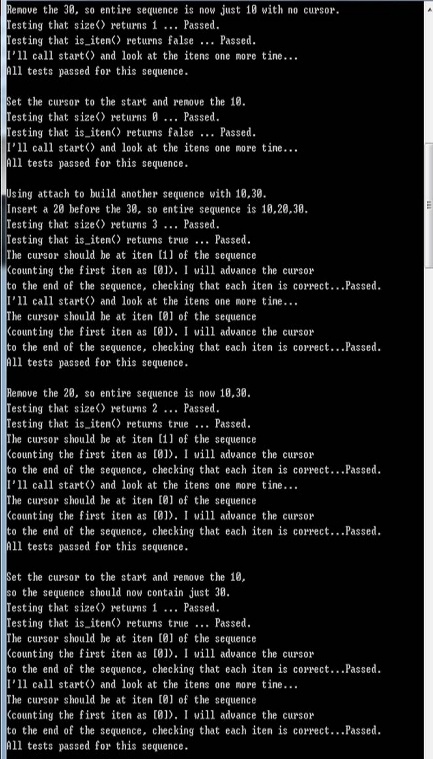
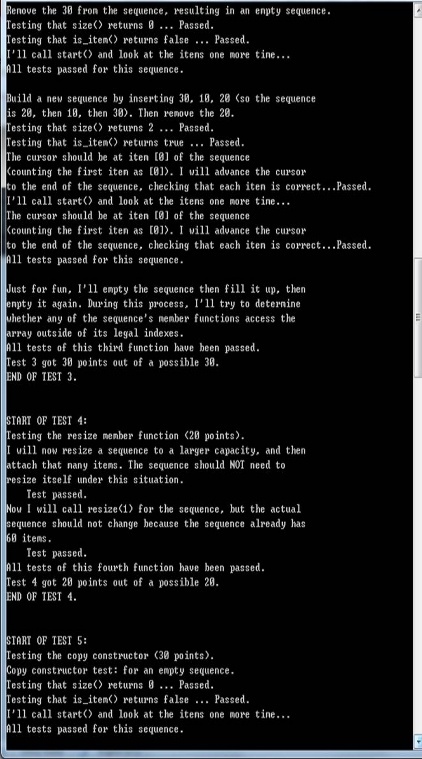
Once again, do your work in small pieces. And using the interactive test program and the debugger to track down errors in your implementation. If you have an error, *do not start making changes until you have identified the cause of the error.*

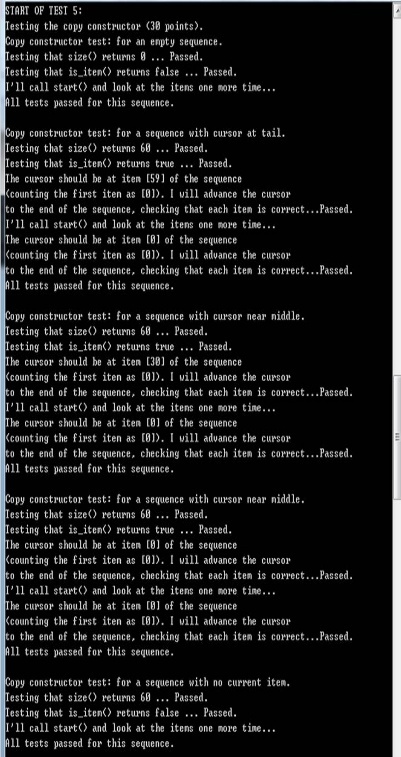
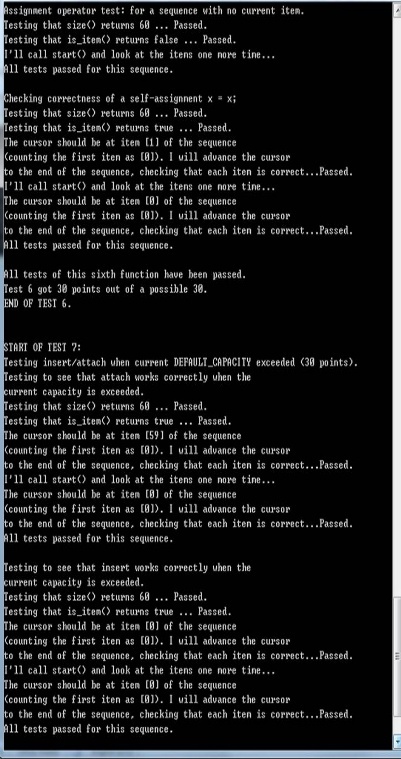
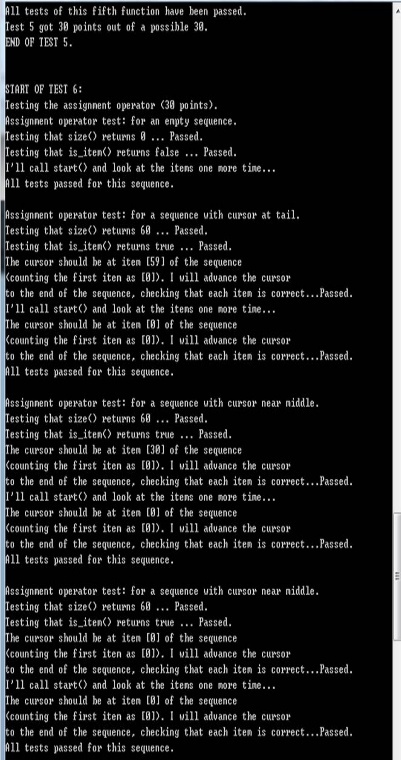
When a member functions needs to increase the size of the dynamic array, it is a good idea to increase that size by at least 10% (rather than by just one item).

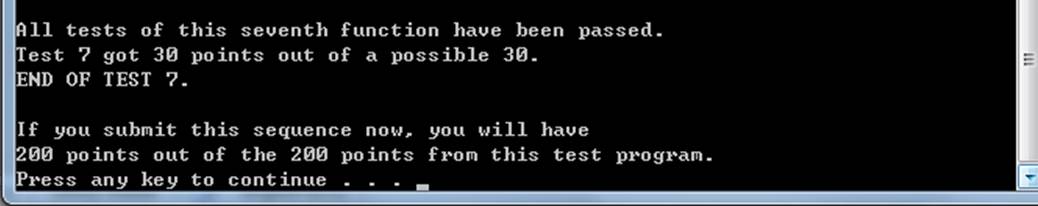
This assignment comes with a CISP430V4A2.zip file. It includes five files (CISP430V4A2Exam.exe, CISP430V4A2Test.exe, sequence\_test.cpp, sequence\_exam2.cpp, and sequence2.h.) The statexam.cpp is a driver file to find out what score you get for this assignment. The stattest.cpp is a driver file to test the program. The sequence2.h file is the declaration file for the sequence2.cpp file you will implement.

The CISP430V4A2Exam.exe, and CISP430V4A2Test.exe are executable files. If you perfectly finish the stats.cpp you can

1. Put the sequence\_exam2.cpp, sequence2.h and sequence2.cpp in a project and run to a result as you double click on the CISP430V4A2Exam.exe. This should tell you what score you get on this assignment. The following picture displays the result.



1. Put the sequence\_test.cpp, sequence2.h and sequence2.in a project and run to a result as you double click on the CISP430V4A2Test.exe. This should help you on your program. The following pictures display the result.

Please document the sequence2.h and sequence2.cpp files properly and zip them into a proper named zip file for the assignment (refer to the assignment section of the class syllabus) and submit it to the A2 folder of the Canvas Website.

Worth 200 points