COM267 Fall 2016
PA4:

Due Date: 02.12.2016, 23:59

The Assignment:

You will implement and test a revised sequence class that uses a linked list to store the items.

Purposes:

Ensure that you can write a small class that uses the linked list toolkit to create and manipulate a linked list.

1. `sequence3.h:` The header file for the new sequence class. Actually, you don't have to write this file. Just start with our version.

   By the way, you might want to compare this header file with your first sequence header file (sequence1.h). The linked list version no longer has a CAPACITY constant nor a DEFAULT_CAPACITY constant because the items are stored on a linked list instead of an array

2. `sequence3.cpp:` The implementation file for the new sequence class. You will write all of this file, which will have the implementations of all the sequence's member functions. You will submit this file.

Other files that you may find helpful:

1. `sequence_exam3.cpp:` A non-interactive test program that will be used to grade the correctness of your new sequence class.
2. `node1.h` and `node1.cpp:` Copy these files to your subdirectory. They contain the linked list toolkit. You may use these files without changing them.

The Sequence Class Using a Linked List

Discussion of the Assignment

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

- The sequence's items are now stored on a linked list. The head pointer of the linked list is a private member variable of the sequence class. I suggest that you also have a tail pointer as an additional private member variable of the sequence class.
- Because you are dynamically allocation memory within your sequence class, you will need to define a copy constructor, an assignment operator, and a destructor.

Once again, do your work in small pieces. For example, at the beginning of your development, start writing your sequence class with a constructor, start, insert, advance, and current member functions. Other member functions can be started out as stubs (i.e. initially implemented as empty functions-just to be able to compile without problems-; later on the implementation is finalized one by one, after validating each implementation).

Use the interactive test program and the debugger tools 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*.

Submission Files:

Submit only your sequence3.cpp file. Your file should be renamed as: **sequence3_<YOUR_ID>.cpp**

Example: Assume that your id is 11290001. Then the file will be renamed as: **sequence3_11290001.cpp**