CS380 Algorithm Design & Analysis
Assignment 2: Insertion and Merge Sort

Date Assigned: Monday, February 2, 2015
Date Due: Monday, February 16, 2015 at 9:15am
Total Points: 50 pts

For this assignment you are to implement both insertion sort and merge sort in C++ using the Visual Studio project (CS380Sorting.zip) that I have placed on the website. We want to implement these sorting algorithms in as general a way as possible, so the code you write will not be specific to a data structure (like an integer array) and therefore can be easily reused. In the code described below, both SortableContainer and SortAlgorithm represent the general, and SortableArray and BubbleSort represent the specific. Note that SortableArray knows nothing of BubbleSort and vice versa.

The Visual Studio Solution contains a number of files:

- ComparableItem.h - an abstract class that defines the interface needed for comparing two objects.
- Student.h/cpp - a concrete class that implements ComparableItem and represents a Student
- SortableContainer.h – an abstract class that defines the interface needed for a SortAlgorithm.
- SortableArray.h/cpp – a class that inherits from SortableContainer and provides an array like container for ComparableItem. This class accepts a SortAlgorithm to use when a user invokes the sort() method.
- SortAlgorithm.h/cpp – an abstract class that defines the interface for a sorting algorithm. This will be used as part of the Strategy design pattern. http://www.oodesign.com/strategy-pattern.html
- BubbleSort.h/cpp – a class that inherits from SortAlgorithm and implements Bubble Sort.
- BubbleDriver.cpp – a simple driver that reads in Students from a file and sorts those students.

To start:

1. Study the Class Diagram to understand the inheritance structure.
    a. Highlight all of the .h files, right click, View Class Diagram.
    b. Right click the class diagram, Layout Diagram.
2. Read the header files to understand the interface for each class.
3. Read the BubbleSort.cpp implementation to understand how a concrete SortAlgorithm is run.
4. Read the BubbleSortDriver.cpp to understand how to use both the SortableContainer and SortAlgorithm.

You must provide:

- **Mountain.h/cpp** – a class that stores information about a mountain (Name, Height). This information will be read from mountains.txt. This must subclass ComparableItem and allow Mountains to be sorted by height.

- **MergeSort.h/MergeSort.cpp** – subclass SortAlgorithm and implement MergeSort.

- **InsertionSort.h/InsertionSort.cpp** – subclass SortAlgorithm and implement InsertionSort.

- **SortingDriverTesting.cpp** – Write a driver that will read all the mountains into a SortableArray and use each of Bubble, Insertion, and Merge sort to sort the array. For each sort routine, sort the array from its original, unsorted order to ascending order. Then, sort this newly sorted array in descending order. Print each of the sorted arrays, ascending then descending. Make sure to reset the array to its original unsorted state before using each sort routine.

What to Submit

You must use Subversion via AnkhSVN inside of Visual Studio. Name your project CS380Sorting_PUNetID. Be sure to follow the coding standards.

In your Visual Studio project you need a text file "CS380Sorting_PUNetID_Answers.txt"

- How many hours did you work on this project?
- Walk through each step of Bubble Sort (to produce a list in ascending order) for the following array of integers. Write the state of the array after each step:
  4 5 2 1 6
- What is the running time of Bubble Sort? Give both the best case and worst case.

You must produce a Class Diagram in Visual Studio:

- Highlight all of the .h files, right click, View Class Diagram.
- Right click the class diagram, Layout Diagram.
- You may need to hand update the layout to fit the diagram on a page of paper. Save this class diagram in your Visual Studio solution and print the diagram to be turned in with your code.

I will pull your project out of Subversion. You must provide me with a color, double sided hard copy of **your code** (not the code I gave you). Make sure you turn on line numbers in Visual Studio.

Your code must build without any warnings. You must follow the C++ coding standards.

Start early!

Translating the Pseudo code in your book for Merge Sort to C++ is non-trivial.

Get the starter project from the class website.
Make sure you rename the folder to be CS380Sorting_PUNetID.

Valgrind for Visual Studio
Valgrind does not work with Visual Studio, however you can use the CrtDebug interface which is built into Visual Studio. You can use the mem_debug.h header file (provided in the Visual Studio solution). Include that file as the first header file in your driver. If you want to turn on memory leak detection include **#define MEM_DEBUG** immediately before including mem_debug.h and add the line
_CrtSetDbgFlag(_CRTDBG_ALLOC_MEM_DF | _CRTDBG_LEAK_CHECK_DF);
as the first line in main() [even before any variable declarations].

See BubbleDriver.cpp for an example.
http://msdn.microsoft.com/en-us/library/vstudio/x98tx3cf%28v=vs.100%29.aspx

Notes:

A well written set of classes will allow you to reuse code later in the semester!

This project will have multiple files with a main() function (BubbleSortDriver.cpp and SortingTestDriver.cpp). Only one main() may be active at a time. To turn off one of the drivers, Right Click on the file | Properties | Excluded From Build | YES/NO. A small red circle should be displayed on the excluded file.

Questions to answer as you review the code:

- What does each **const** mean? Why are they necessary?
- Moving from Bubble Sort to Merge Sort to Insertion Sort, what changes? How does the design handle this change elegantly?
- What is **static casting**? What is **dynamic casting**? How do they differ? Why is each important or necessary?
- Why are **virtual destructors** used?
- Why are methods marked **virtual**?