CS380 Algorithm Design & Analysis
Assignment 3: Quick Sort and performance measurement

Date Assigned: Sept 30, 2013 **Date Due: October, 11, 2013: 4:45 pm**

Total Points: 50 pts

For this project you will implement QuickSort in the same manner as MergeSort and InsertionSort in project 1. You may add this code to your Sorting Project.

You will also need to write a new driver that will run each of these four (merge*****, insertion, quick) algorithms and time the execution of each sort routine and track how many times needSwap is called by each sort. Three new files, largeMountains.txt, largeMountainsASC.txt, and largeMountainsDESC.txt is provided. These each contain 1,000,000 randomly generated mountains.

*****Also, reimplement MergeSort and use either a native C array or a SortableArray for your temporary storage, which ever you did not do in the previous assignment. When you turn in your assignment you must have**two** working implementations for merge sort.

Your driver must, for each of the three files:

Read the first 100 mountains, and sort using each of the four algorithms. Time the call to sort() in each case and track the number of needSwap()s called. Sort in the DESC direction. For each sort algorithm, reload the data from the file before sorting.

Run Insertion sort for each of the sizes 100, 1,000, 10,000, and 100,000. Run Merge sort and Quick sort for each of the sizes 100, 1,000, 10,000, 100,000, and 1,000,000.

OUTPUT

```
Filename: largeMountains.txt
Size 100
Insertion Sort:
        Swaps: XXXX
        Time:  XXXX
Merge Sort: Native Array
        Swaps: XXXX
        Time:  XXXX
Merge Sort: SortableArray
        Swaps: XXXX
        Time:  XXXX
Quick Sort:
        Swaps: XXXX
        Time:  XXXX

<repeat for each size>
```

What to Submit

I will pull your project out of Subversion. You must provide me with a color, double sided hard copy of

QuickSort.h / QuickSort.cpp
PerformanceDriver.cpp
<any other NEW or CHANGED source files>
A printout of the text file containing your answers to the questions**posted ON MOODLE.**
A print out of your output

Your code must build without any warnings. You must follow the C++ coding standards. Check for memory leaks!

Start early! **THIS MAY TAKE HOURS TO RUN, PLAN ACCORDINGLY.**

Get the files at:
http://zeus.cs.pacificu.edu/chadd/cs380f13/largeMountains.zip

You may have many, many calls to needSwap.
http://msdn.microsoft.com/en-us/library/s3f49ktz(v=vs.90).aspx

Hints on using timers in C++:

```
#include <ctime>
clock_t start, finish;
start = clock();
sort(); // Call your sorting algorithm
finish = clock();
cout << "Time for sort (seconds): " << ((double)(finish -
 start))/CLOCKS_PER_SEC;

OR

#include <windows.h> //and follow this link
http://stackoverflow.com/questions/1739259/how-to-use-
queryperformancecounter/1739265#1739265
```

Be sure to do all of you timing via “Run without debugging” and without memory debugging.

For deeply recursive algorithms, you may need to increase the available statck space for your project.

Properties | Configuration Properties | Linker | System |

Stack Reserve Size 8388608 (number of bytes)

Stack Commit Size 4000000

Bonus:

Perform timings for Heap Sort as well for whatever sizes seem reasonable. For heap sort, you must insert each item into the heap AND extract each item. The timed operation is the extraction of all elements.