CSS 342 – Project Five

Pedagogical goals:
Get an instinctive understanding of how different sorting algorithms perform at load.
Introduce the idea of monitoring what your program is doing.
Give you practice in various sorting algorithms.

You will implement an ArrayList class deriving from the provided ListInterface class.

You will add additional instrumentation to your ArrayList class that will return the number of times a particular method has been called. See header file. Note that the statistic variables must be static. IE when you make a new array as part of merge sort, any manipulation to the new array must count in your statistics.

A test program (arrayListTest) is in the files section.

You will write a program to read in a data file, and an optional command from the command line. The data file will contain integer values in a random order, one value per line. The optional command '-d' will tell the program to dump the sorted output of the file to cout (see format below);

So "sortstatistics -d filename" will read in the file, and print out the sorted file. "sortstatistics filename' will print out the statistics, but not the sorted file.

You will read in the data file and sort it using a Insertion Sort, Merge Sort, and Quick Sort. (Note: be sure to start from an unsorted array each time).

After you have done your two sorts you will print out a report in the following format (no -d option)

Filename: <filename>
Number of items: <num items>

Number of	Access	Swap	Remove	InsertAt	Append
InsertionSort	<N>	<N>	<N>	<N>	<N>
MergerSort	<N>	<N>	<N>	<N>	<N>
QuickSort	<N>	<N>	<N>	<N>	<N>

Printing hint:
#include <cstdio> and create a line of output with fixed number widths using something like

```
char line[80];
sprintf_s(line, "InsertionSort: %10d %10d %10d %10d %10d", insertionStats.access,
insertionStats.swap, insertionStats.remove, insertionStats.insertAt,
insertionStats.append);
```

If the -d option is selected – you will output the following

Filename: <filename:
Number ofitems: <Num items>
Insertion Sort Results:
N1 N2 N3 …… NN
MergeSortResults:
N1 N2 N3 …… NN
QuickSortResults:
N1 N2 N3 …… NN

Number of	Access	Swap	Remove	InsertAt	Append
InsertionSort	<N>	<N>	<N>	<N>	<N>
MergerSort	<N>	<N>	<N>	<N>	<N>
QuickSort	<N>	<N>	<N>	<N>	<N>

Note – be sure to reset your instrumentation before you start the searching (the instrumentation values must be static)

Name your program files arraylist.h and sortstatistics.cpp