CS 101 **Project 2** **Summer 2016**

Write a C++ program project2.cpp that operates as described below.

Your program should accept one command line argument, which denotes the array size n. Your program will initialize six arrays of size n in six different ways as described below, and then sort each array using five different sorting algorithms.

Initialize the six floating-point arrays of size n as follows:

- Identical [k] = 0.0
- Ascend [k] = 1.0 * k
- Descend [k] = –1.0 * k
- RevDigit [k] = decimal point followed by digits of k in reverse order, so RevDigit [149] = **.**941
- NegRevDigit [k] = – RevDigit [k]
- Random [k] = floating-point value obtained using a random number generator

Sort each array using these five sorting algorithms:

- Insertion sort
- Selection sort
- Bubble sort
- Merge sort
- Quick sort, using middle element A[(low+high)/2] as pivot value

During each sorting algorithm, count the number of these operations that are performed:

- Comparisons: operations <, >, ≤, ≥ that involve array elements
- Swaps: exchanging the locations of two array elements
- Assignments: modifying an array element, other than during initialization or swaps

Your program's output will summarize the total number of these operations (comparisons + swaps + assignments) performed by each of the five sorting algorithm using each of the six input arrays.

Hints: All five sorting algorithms will perform comparison operations. The four in-place sorts (that is, all except merge sort) should do swap operations. Only merge sort needs assignment operations. If your counts are larger, you should try to implement the sorting algorithms more efficiently, with fewer operations. But if your counts are smaller, you should make sure that your program is indeed counting all the operations that it actually performs.

Here are two example output files for array sizes n=100 and n=5000.

Output100.csv

```
Sort100,Insertion,Selection,Bubble,Merge,Quick
Identical,99,4950,99,1660,1146
Ascend,99,4950,99,1700,771
Descend,9900,5000,9900,1660,822
RevDigit,4149,4995,6822,1873,1203
NegRevDigit,5931,4995,7875,1843,1111
Random,5285,5045,7509,1886,1085
```

Output5000.csv

```
Sort5000,Insertion,Selection,Bubble,Merge,Quick
Identical,4999,12497500,4999,153420,99410
Ascend,4999,12497500,4999,155620,66807
Descend,24995000,12500000,24995000,153420,69308
RevDigit,12424999,12502482,18696622,178101,105111
NegRevDigit,12579968,12502480,18785000,176927,106077
Random,12362164,12502494,18672431,178913,98760
```

The above comma-delimited output files can also be opened with a spreadsheet program such as Microsoft Excel, in which case they will be much easier to read:

output100.csv

Sort100	Insertion	Selection	Bubble	Merge	Quick
Identical	99	4950	99	1660	1146
Ascend	99	4950	99	1700	771
Descend	9900	5000	9900	1660	822
RevDigit	4149	4995	6822	1873	1203
NegRevDigit	5931	4995	7875	1843	1111
Random	5285	5045	7509	1886	1085

output5000.csv

Sort5000	Insertion	Selection	Bubble	Merge	Quick
Identical	4999	12497500	4999	153420	99410
Ascend	4999	12497500	4999	155620	66807
Descend	24995000	12500000	24995000	153420	69308
RevDigit	12424999	12502482	18696622	178101	105111
NegRevDigit	12579968	12502480	18785000	176927	106077
Random	12362164	12502494	18672431	178913	98760

Because one of the input arrays contains randomly generated values, the number of operations performed by each sorting algorithm on this array may vary slightly each time the program runs.

Your program should write to standard output stream. Download the example files provided to see the precise output formats.

Please carefully read the following requirements:

- You must do your own work; you must not share any code. If you violate this rule, you may receive an invitation to the dean's office to discuss the penalties for academic misconduct.
- Make sure your program runs properly on cs-intro.ua.edu. Your program will be graded on that system.
- Submit your project in a zipfile that contains all your source files (.cpp and .h files) that are needed to build the executable program. Optionally you may also include a makefile. There should be no subdirectories or extra files.
- If you violate the requirements such that it breaks our grading script, your project will be assessed a significant point deduction, and extreme or multiple violations may cause the project to be considered ungradable.
- Three pairs of example output files and a test script are provided. Run the following commands to test your program. If any differences appear (other than slightly different numbers on the "Random" lines), then your project does not match the specifications. Points will be deducted for each incorrect line of output.

```
unzip *.zip
chmod u+x testscript.sh
./testscript.sh
```

- Alternatively, instead of running the test script, you can instead enter these commands.

```
g++ project2.cpp –Wall –lm –o project2
./project2 3 > temp3.csv
diff output3.csv temp3.csv
./project2 100 > temp100.csv
diff output100.csv temp100.csv
./project2 5000 > temp5000.csv
diff output5000.csv temp5000.csv
```

- Submit your project on Blackboard by the due date (11:59pm Friday). There is a grace period of 24 hours (until 11:59pm Saturday). Projects submitted on Sunday will be assessed a late penalty of 5% per hour. No projects will be accepted after Sunday.
- Double-check your submission when you submit it. Errors discovered later cannot be fixed and resubmitted after the project is graded. Projects will not be re-graded unless an error is found in the grading script or in the input/output files used during grading.