**MSCF**

**Financial Computing I**

**Homework 5, for Lectures 15 through 20**

***Due At 12:30 pm, Tuesday, Oct. 2, before Lecture***

***You will lose 1 point every 10 minutes after that time***

1. **Sorting Algorithm Performance**
2. Create a new, empty project named **hw5.1**. Add the file **hw5.1.cpp** to the project. This source code file contains C++ implementations of the seven sorting algorithms we have discussed (selection, insertion, bubble, “optimized” bubble, quick, merge, and counting), with very brief tests of each of them. It also contains a timing test for a selection sort of 10,000 “random” **int** values (we have used the **rand()** function to generate these 10,000 values; rand() has some problems, but it is okay for this purpose). Compile and test.
3. Add timing tests for the other six sorting algorithms, and test each one with n\_value doubling from 10,000 up to 320,000 (or even higher). In your code, create a table in a comment block showing the number of seconds that each sorting algorithm took. You may STOP timing any sorting algorithm that exceeds 2 minutes (120 seconds) of time.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Time** (seconds) | **n\_values**  10,000 | **n\_values**  20,000 | **n\_values**  40,000 | **n\_values**  80,000 | **n\_values**  160,000 | **n\_values**  320,000 |
| **selection** | 22 | 88 | 380 |  |  |  |
| **insertion** | 13 | 52 | 220 |  |  |  |
| **bubble** | 62 | 241 |  |  |  |  |
| **bubble (opt)** | 34 | 135 |  |  |  |  |
| **quick** | 0 | 0 | 0 | 0 | 0 | 0 |
| **merge** | 0 | 1 | 1 | 2 | 4 | 8 |
| **counting** | 0 | 0 | 0 | 0 | 1 | 1 |

Do the sorting algorithms behave approximately as expected in time performance? Make a comment in your code.

***REMEMBER*** to put all authors’ names into each of your source code files.Put your **hw5.1.cpp** file into a **.zip** archive and upload to the course web site.