**CS&E A311, Spring 2020**
**Assignment 1**
**20 points**
**Due 24 Jan 2020 at 11:59pm, 1 document submitted to Blackboard, 3 files to *transformer***

This assignment will ensure that you have access to the class linux server, *transformer,* uaa-transformer.duckdns.org (IP address: 137.229.181.88), have basic Unix OS skills to create directories and files, and can compile your code on the server.

For this assignment, create a directory in your home directory called `a1`.

1. (2 pts) In this folder, create a text file called `a1.txt` using a Unix text editor of your choice (e.g. nano, emacs, vi/vim) and type a sentence in the file telling me which editor you used to create the file.
   Type 'man' before each of the following commands, and then run each of them separately and experiment with one or more arguments:
   `ls, mkdir, cd, more, less, cat, man, apropos, javac, g++, rm, mv, ps, df`
   Once you are finished, add a line to your a1.txt file stating that you have at least a basic understand of each of the commands.
2. (8 pts) In the same folder, write a simple C++ function that implements Insertion Sort in C++ (filename `InsertionSort.cpp`). Use the pseudo-code in Chapter 1 of the Cormen et al. textbook (same as the pseudo-code on the class notes). From your main method, test your implementation on arrays of length 0, 10 (print the before and after sorting values to the console), and randomly generated arrays of length 100, 1000, and 10000 numbers. Conduct an empirical complexity analysis by adding a counter within each loop (like the pseudo-code in presented in class). Print the values of the inner and outer counter to the console. Something like:

```
For an array of length 100, the outer loop ran xxx times, and the
inner loop ran yyy times.
```

3. (5 pts) Consider sorting *n* numbers stored in array A by first finding the smallest element of A and exchanging it with the element in A[1]. Then find the second smallest element of A, and exchange it with A[2], and so on. Write pseudocode for this algorithm. Give the best-case and worst-case run times using Theta notation. Justify your answers similar to the Insertion Sort analysis presented in class. Submit your answer to this question to **Blackboard** as a document (e.g. .docx/.pdf file) or scan/photo of your hand-written work (e.g. .jpg file).
4. (5 pts) Write a program in C++ (filename `Sort.cpp`) that implements the above sorting algorithm. Run the algorithm on a 10-element integer array (that you create) and print the array to the standard output at each step of the outer loop to show the sorting process.

You may develop your C++ code on any platform, but be sure your programs run on *transformer* using the g++ compiler. Use good coding style and documentation for this and all programming assignments.

Please be sure to add your name and date to all 4 files.

I will copy the contents of your a1 directory at the due date and time.