ECE2036 **Springl Semester, 2016**

Project 1 – Bubble Sort Timing Analysis

Assigned: January 20, 2016 *Due: January 27, 2016*

Introduction. For this assignment, we will implement a simple "bubble sort" algorithm and measure it's execution time over several different array sizes to be sorted. There is quite a bit of "skeleton" code provided for you in the `bubble-sort-skeleton.cc` file which is given to you as a starting point. The skeleton code calls the BubbleSort function (that you write) using array sizes of 100, 1000, 10000, 100000 integers.

The bubble sort algorithm is known to have a runtime complexity proportional to N^2, where N is the number of elements to be sorted. Given that, the measured execution of each sort iteration should be 100 times slower than the previous, since each of the array sizes increases by a factor of 10.

The skeleton code uses a function called `ticks()` that reads the CPU internal clock cycle counter and returns a 64–bit value. The main program simply reads the cycle counter before and after the BubbleSort function, and subtracts to compute elapsed time in units of CPU ticks. If your implemention of the BubbleSort is correct, the elapsed tick count for each iteration should be 100 times the tick count of the previous

The skeleton code then calls `CheckIt` which verifies your sorted array is indeed properly sorted.. If the "Error-Count" value reported by th`CheckIt()` function is not zero your BubbleSort is buggy and you must fix it.

Specific Program Requirements.

1. You must implement the `BubbleSort` function which is included in the skeleton code but not implemented.
2. Once you have it implemented and compiled with no errors or warnings then run the program, observing the reported Execution time in ticks.

Resources. There are a number of files that are given to you. These are all on the `deepthought19` system, and you will copy these to your own directory (instructions as to how to do this are below).

1. A skeleton `bubble-sort-skeleton.cc` program that is a starting point. It contains the main program and several other useful functions.
2. `check-it.h` and `check-it.cc` define and implement the function to verify your array is properly sorted.
3. `ticks.h` and `ticks.cc` define and implement the CPU cycle counter read.
4. A `Makefile` which you can use to compile the program.

Your program can be compiled and tested on any available computing platform that has a C++ compiler. The instructor and TA will compile and test your program on the deepthought19linux systems. Be sure to put your name on the source code in the comments section.

Copying the Project Skeletons

1. Log into `deepthought19.cc.gatech.edu` using `ssh` and your prism log-in name.
2. Copy the files from the ECE2036 user account using the following command:

   ```
   /usr/bin/rsync -avu /nethome/ECE2036/BubbleSort .
   ```

   Be sure to notice the period at the end of the above command.
3. Change your working directory to `BubbleSort`

```
cd BubbleSort
```

4. Copy the provided `bubble-sort-skeleton.cc` to `bubble-sort.cc` as follows:

   ```
   cp bubble-sort-skeleton.cc bubble-sort.cc
   ```

5. At this point, you can run the `make` program to compile the skeletons. Of course it won't do anything useful since you have not implemented your `BubbleSort` algorithm. To run the `make` program simply type `make` in your terminal window after having changed your working directory to the BubbleSort directory.

6. You can edit your program using any of the available text editors found on deepthought, specifically `vi`, `vim`, and `emacs`. There are plenty of on-line resources about using text editors on linux.

Turning in your Project. The system administrator for the cluster has created scripts that you are to use to turn in your project. The script is called `turnin-ece2036a` (or `turnin-ece2036b` depending on which section you are in). The script is found at and is found in `/usr/local/bin`, which should be in the search path for everyone. From your **home directory** (not the BubbleSort subdirectory), enter:

```
turnin-ece2036a BubbleSort
```

This automatically copies everything in your `BubbleSort` directory to a place that we can access (and grade) it.