Version: 9/16/2013

ECE/ME/EMA/CS 759
High Performance Computing for Engineering Applications
Assignment 2

Date Assigned: September 16, 2013
Date Due: September 23, 2013 (11:59 PM)

1. There are two related functions that copy/move memory around in C: `memcpy` and `memmove` (assume neither of these functions is an alias of the other). When should you use one or the other? Are there any performance issues that should dictate your choice?

2. Write a program that reads in a set of words/numbers provided as command line arguments. Within a function that you come up with, count the total number of characters in the input excluding white spaces. Return this value and print it out in the `main` function. Note: the set of words/numbers can contain combinations of letters and digits only. Don't use other special characters. Here's a possible scenario:

```
>> myProgram this is a test 4 you 2
```

The output of the program should be 16, since there are 16 characters in the input provided above (excluding white spaces). For this problem you might want to use the variables `argv` and `argc` that you have access to in the `main` function of your program.

3. Write a C or C++ program that takes as an argument an input file, it opens the file, reads all the integer numbers (no more than 1,000,000 of them) and sorts them from the smallest to the largest. The input file contains one integer per line.

   a. Without consulting any online source or book, use an algorithm that you design or are familiar with to generate a program that prints out four things: (*i*) the number of integers read in, (*ii*) the smallest integer read in, (*iii*) the largest one, and (*iv*) the amount of time in milliseconds (ms) required to sort these integers.
   b. In order to gauge the performance of your sort algorithm consult a book, online source, or use a library that has a sort function implemented to sort the same collection of integers as at a) above to verify your answer.
   c. To understand the behavior of the two approaches at a) and b) above, run them on a set of nine input files containing 2^{10}, 2^{11},…, 2^{19} integers generated randomly. Record the time in ms (milliseconds) and run a standard regression to approximate the curves that give ordering time versus number of integers for the two algorithms. This is called a scaling analysis. Provide the plot in **`png`** format and indicate the regression function. Feel free to post this plot in case you want to help other colleagues get an idea of what they should obtain at the end of this exercise.

4. This problem is identical to Problem 3, except that instead of the sorting operation you have to perform an exclusive scan operation. For a definition of the scan operation read pages 1 through 3 of the Blelloch 1990 paper available on the class website. Your program should be able to take one input argument which is the name of the text file storing the integers. The input file contains one integer per line.

   a. Implement an algorithm that executes an *exclusive scan* operation and prints out three things: (*i*) the number of integers read in, (*ii*) the last entry in the scanned array, and (*iii*) the amount of time in milliseconds (ms) required to perform the scan operation.
   b. Run a scaling analysis much like you did for 3.c above but this time only for your algorithm. Test your code with an input file with 2^5, 2^6,…, 2^{12} integers randomly generated with values between -10 and 10. Produce the plot and provide the regression function in the readme file.

FURTHER NOTES ON YOUR HOMEWORK:

- It doesn't really matter how slow your algorithms run in Problems 3 and 4. The point is to understand how difficult it is to design and implement fast algorithms.
- If your code is so fast that it basically reports 0 ms, simply include that value in the plot you were asked to generate. Performing the timing will turn out to be more challenging than expected, particularly if you use Windows specific functions that are not supported under Linux on Euler. When you try to compile your code in Linux you might have some unpleasant surprises. Finally, kudos to you, if you can provide timing resolution at a level below milliseconds; i.e., higher resolution.
- When you generate your plots you don't have to do that within your program. Simply use MATLAB or some **`python`** utility to generate plots at the end, as a separate post-processing step.
- Please make sure your directory contains a readme.pdf file that contains the material that you want the grader to read.
- You are encouraged to use the ME759 Forum for exchanging ideas or sharing links to helpful information, but no posting of the solutions please.
- This set of problems is similar to what was assigned a year ago. You might want to quickly read through the Forum posts for ME964 of Spring 2012. There are some valuable comments provided therein.