CSE 603 **Programming Assignment 1**
Due Date: Oct 7, 2014
Weight: 7% of your grade

Problem: Simple Prime Number Generator using OpenMP, MPI, and SSE/AVX: study of scalability.

Description: In this assignment you will optimize parallel programs using OpenMP in a shared-memory environment, using MPI in a distributed-memory environment, and using SSE/AVX in a vector-processing environment. In particular, you will have to parallelize a simple prime number generator. This program is very simple and is intended to get you familiar with OpenMP, MPI and SSE/AVX.

Start with a uniprocessor (sequential) program written in C for determining prime numbers.
The program works by testing each odd number (up to a specified limit) for divisibility by all of the factors from 3 (you already know that 1 and 2 are prime numbers) to the square root of that number. This is a simple algorithm that is easy to parallelize but not a good algorithm to determine prime numbers.

- Your task is to parallelize the algorithm in a number of different ways using the OpenMP API on the shared-memory nodes of CCR.
- The program takes two main parameters, which are read in from the command line:
  P: the number of processors and N: the problem size.
- In addition, the program takes arguments that output all of the primes generated, either to a file or standard out. This should assist you in ensuring that the parallel version of the algorithm works correctly.

What you need to do:

- Your report should include the following:
  - Your rationale for parallelizing the program
  - The largest prime number and number of prime numbers you were able to compute.
  - Speedup plots for varying size of N and P.
  - Complete source codes in different directories – sequential, OpenMP, MPI, SSE/AVX, OpenMP+MPI including makefiles for each embedded within the directories. Include scripts you may have used to run programs with different parameters and also scripts that you may have used to collate results and create graphs (highly recommended to do this to make it easier to finish your work in time).
  - Script (with complete description) to run your programs (including samples)
  - Discussion of results to include the following (graphs)
    - Impact on the performance of the algorithm based on
      - Increase in the size of the problem
      - Increase in the number of cores
  - How does this relate to your theoretical evaluation of the scalability of this algorithm?

Extra Credit (2%): Incorporate a combination of OpenMP and MPI and compare the performance in terms of increase/decrease of speedup and scalability.