EE w382V: Multicore Computing
Homework 3

Instructor: Professor Vijay Garg (email: garg@ece.utexas.edu)
TA: Wei-Lun Hung (email: wlhung@utexas.edu)

Deadline: July 9^{th}, 2015

The source code must be uploaded through Canvas before the end of the due date (i.e., 11:59pm on July 9^{th}). The assignment should be done in teams of two. You should use the templates downloaded from the course github (https://github.com/kinmener/UT-Garg-Multicore.git). You should not change the file names and function signatures. In addition, you should not use package for encapsulation. Please zip and name the source code as [EID1_EID2].zip.

1. **(50 points)** The purpose of this assignment is to learn OpenMP. To setup the API, you can install gcc-4.7 compiler on the Linux machine or Visual Studio 2008–2010 C++ on the Windows machine. For more information, please visit: `http://openmp.org/`.

   (a) Write a C/C++ program `MatrixMult` that allows parallel multiplication for two matrices of doubles by using OpenMP. The task for your program is described below:

   Your program should accept three arguments. The first two arguments are paths of the input files that encode two matrices that need to be multiplied. The format of an input file is defined as follows: each input file contains one matrix. The first line of an input file contains two positive integers: m and n denoting the number of rows and columns in the matrix. The next m lines in the file provide rows of the matrices with entries separated by space. The third argument to your program, T, indicates the number of threads to be used. Suppose your program is named `matrix_mult`, and is executed with the following parameters:
   `./mutrix_mult mfile1 mfile2 T`

   The output of your program (to standard output) should be a matrix in the format used for input matrices. Assume that matrices are of the proper form and can be multiplied. Submit a plot of time taken to compute the product of matrices of size 100 by 100 when the number of threads are varied from 1 to 8.

   (b) Write a multithreaded C/C++ function `MonteCarloPi` that calculates the value of π using the Monte Carlo Method. The function returns the value of π as a `double` and its signature is given as follows:

   ```
   double MonteCarloPi(int s)
   ```

   Imagine that we have a square with side length $2R$ and it contains a circle of radius R (see Figure 1). The idea of Monte Carlo Method is that if you randomly choose s points (black and grey dots) inside the square, there are c points (black dots) that are located in the circle, where

$c/s = \pi/4$. In your implementation, you can choose your own value of R and use the equation, $x^2 + y^2 < R^2$, to check if the chosen points are located inside the circle. Your function should choose points in parallel.

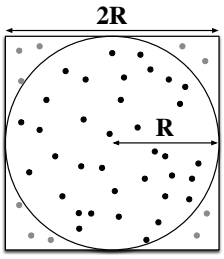

Figure 1: Black and grey dots are the randomly chosen points.

2. **(50 points)** Implement the Anderson's (array-based) and MCS algorithms for lock. Compare the total time taken for each of the following methods for n = 1..6 and m = 1,200,000. Please include your figure in the zip file. Your figure can be drawn using any of your favorite plotting methods.