OS Assignment 3

Course Name:	Operating System Concepts
Topic:	Multithreaded Programming
Programming Language:	Any One
Experimental Environment:	Linux OS
Deadline:	August 17, 2017
Time:	During Class Hour

Problem 1: Write a multithreaded program that calculates various statistical values for a list of numbers. This program will be passed a series of numbers on the command line and will then create three separate worker threads. One thread will determine the average of the numbers, the second will determine the maximum value, and the third will determine the minimum value. For example, suppose your program is passed the integers

90, 81, 78, 95, 79, 72, 85, 46, 78, 32, 86, 96, 55

The program will report

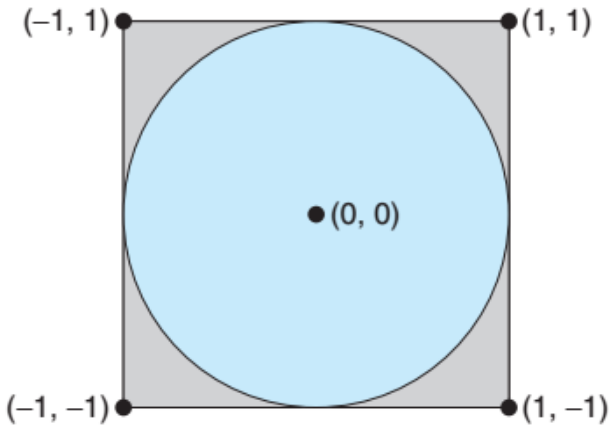
```
The average value is 74.84

The minimum value is 32

The maximum value is 96
```

The variables representing the average, minimum, and maximum values will be stored globally. The worker threads will set these values, and the parent thread will output the values once the workers have exited. (We could obviously expand this program by creating additional threads that determine other statistical values, such as median and standard deviation.)

Problem 2: Write a multithreaded program that outputs prime numbers. This program should work as follows: The user will run the program and will enter a number on the command line. The program will then create a separate thread that outputs all the prime numbers less than or equal to the number entered by the user. The numbers of created threads are equal to the square root of inputted number.



Problem 3: An interesting way of calculating is to use a technique known as ***Monte Carlo,*** which involves randomization. This technique works as follows: Suppose you have a circle inscribed within a square, as shown in above Figure (Assume that the radius of this circle is 1.) First, generate a series of random points as simple (x, y) coordinates. These points must fall within the Cartesian coordinates that bound the square. Of the total number of random points that are generated, some will occur within the circle. Next, estimate π by performing the following calculation:

π = 4× *(number of points in circle)* / *(total number of points)*

Write a multithreaded version of this algorithm that creates a separate thread to generate a number of random points. The thread will count the number of points that occur within the circle and store that result in a global variable. When this thread has exited, the parent thread will calculate and output the estimated value of π. It is worth experimenting with the number of random points generated. As a general rule, the greater the number of points, the closer the approximation to π. In the source-code download for this text, we provide a sample program that provides a technique for generating random numbers, as well as determining if the random (x, y) point occurs within the circle.