CSC 428 – Operating Systems

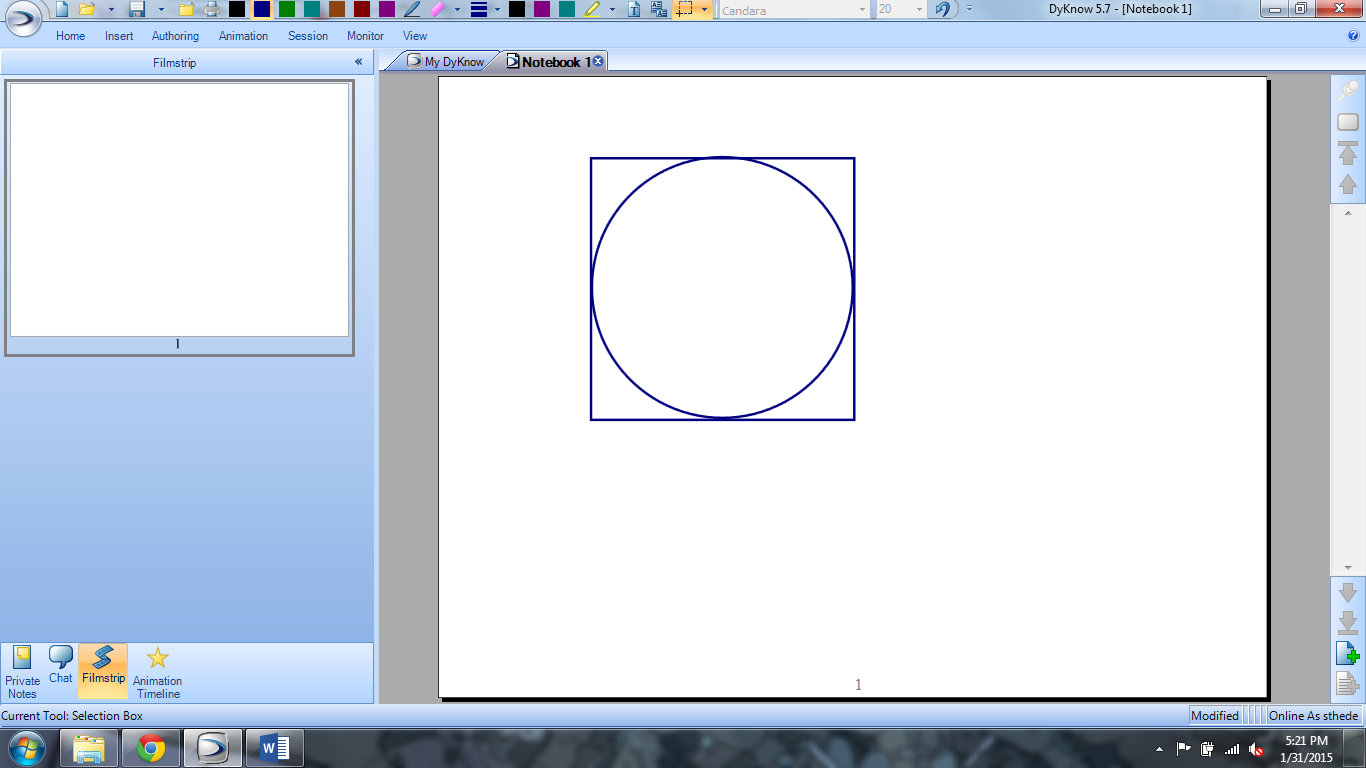
**Programming Assignment #3 – Parallel Monte Carlo Simulations**

.

*The Assignment:*

This assignment deals with writing a ***Monte Carlo*** simulation. Monte Carlo methods use random number generators to simulate other real-world activities. For example, you could study the behavior of the stock market by generating random numbers to simulate investors buying and selling stocks. They are specifically valuable when more conventional analysis is difficult or impossible.

The application we will be coding is a relatively simple application to statistically estimate the value of π. Remember that π is the ratio of a circle’s perimeter to its diameter (or the ratio of its area to its radius squared). To do this estimation, imagine you have a square of fabric two feet to a side. Within this fabric (exactly in the center) you place a dartboard with a radius of one foot. This should look something like this:



Now close your eyes and throw darts at the dartboard / fabric. Assume that the darts will all land on the fabric, and that they are evenly (and randomly) distributed across the square as well. Then, consider the number of darts that hit the dartboard versus the total number of darts thrown. Assuming that the darts are thrown randomly, we get something like this:

So, if we take the number of darts hitting the dart board (the circle), multiply it by 4, and divide it by the total number of darts thrown, we should get an estimate of π.

*Programming Details:*

We can simulate the throwing of darts by generating random numbers. Specifically, we generate two random numbers (real numbers) that range from -1 to 1. These numbers represent the coordinates of the location where the arrow hits. Then, determine if that location lies within the circle by calculating its distance from the origin – if it is 1 or less units away, it is in the circle. Repeat a large number of times, and calculate π from the results.

We will use multiple threads to do this process several times in parallel, then combine the results to determine a final value for π. Specifically, your program should do the following:

* Inform the user what the program will be doing.
* Have the user enter a number of threads to run – this number should not be larger than 20 (and no less than 1, obviously).
* Have the user enter a number of arrows to “throw” in EACH thread – this number can be any positive number.
* Execute each thread concurrently. Each thread should calculate its own value of π and pass it back to the main program (see hints).
* The main program should calculate a final value of π by averaging the results from all the threads. Report this result to the user.

Finally, you should create a table of data that reports the results of running your program in all combinations of the following:

* 1, 2, 5, 10, and 20 threads
* 10, 100, 1000, and 10000 arrows in each thread

So, you should report a total of 20 different values of π. Along with your table of data, provide a paragraph or so of discussion of the results you found.