ECE 2036 Lab 6
Multi-threading using std::thread

(100 pts)

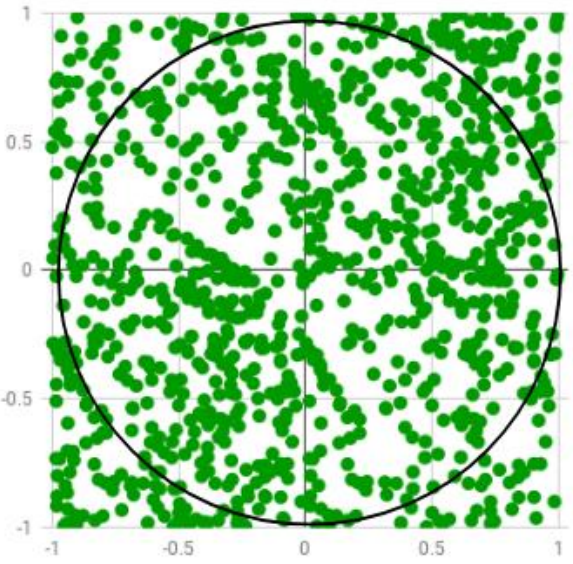
Due: April 21, 2020 by 11:59 PM

This lab is intended to help you explore the advantages and disadvantages of using multiple threads in your applications. We will be using the classic estimation of $\boldsymbol{\pi}$ with a Monte Carlo simulation to better understand how multi-threading can be used to improve computation time.

Estimating π using a Monte Carlo simulation:

(https://www.geeksforgeeks.org/estimating-value-pi-using-monte-carlo/)

"The idea is to simulate random (x, y) points in a 2-D plane with domain as a square of side 1 unit. Imagine a circle inside the same domain with same diameter and inscribed into the square. We then calculate the ratio of number points that lied inside the circle and total number of generated points."



The area of the square is 1 sq unit and the area of the inscribed circle is: $\pi r^2 = \pi \left(\frac{1}{2}\right)^2 = \frac{\pi}{4}$ sq units.

$$\frac{\text{area of the circle}}{\text{area of the square}} = \frac{\text{no. of points generated inside the circle}}{\text{total no. of points generated or no. of points generated inside the square}}$$

that is,

$$\pi = 4 * \frac{\text{no. of points generated inside the circle}}{\text{no. of points generated inside the square}}$$

Assignment:

Create a console application that:

1) Informs the user of the maximum number of concurrent threads:
2) The program then continuously prompts the user for the following information shown below and outputs the processing time in microseconds followed by a std::endl. If the number of threads is equal to zero the program ends.

> 8 concurrent threads are supported.
>
>Please enter the number of threads (0 to exit): 1
>Please enter the number of calculations for each thread: 1000000
>Calculated value of pi: 3.27138
>Processing time (microseconds): 148228
>
>Please enter the number of threads (0 to exit): 8
>Please enter the number of calculations for each thread: 100000
>Calculated value of pi: 3.27511
>Processing time (microseconds): 40773

>Please enter the number of threads (0 to exit): 0

Remember to use your main thread as one of the requested threads. If a single thread is requested then only the main thread is used. If 10 threads are requested then 9 std::thread objects should be created.

Remember to comment your code. Normal commenting standards apply.

You need to create a function that takes as an input parameter the number of calculations to perform and then updates an atomic variable of type **unsigned long long** to track the number of calculations that were inside the circle. Use a local variable to only update the atomic variable once for each thread.

```
std:: atomic_ullong ulNumInside (0);
```

void RunCalculations(unsigned long long ulNumCalculations)
{
// Use a local variable to keep track of results inside circle and only update the atomic variable at the end
}

You can use parts of the code from (https://www.geeksforgeeks.org/estimating-value-pi-using-monte-carlo/) to help create your program.

ECE 2036 Lab 6 Turn-in Sheet

Use your program on coc-ice.pace server to answer the following questions:

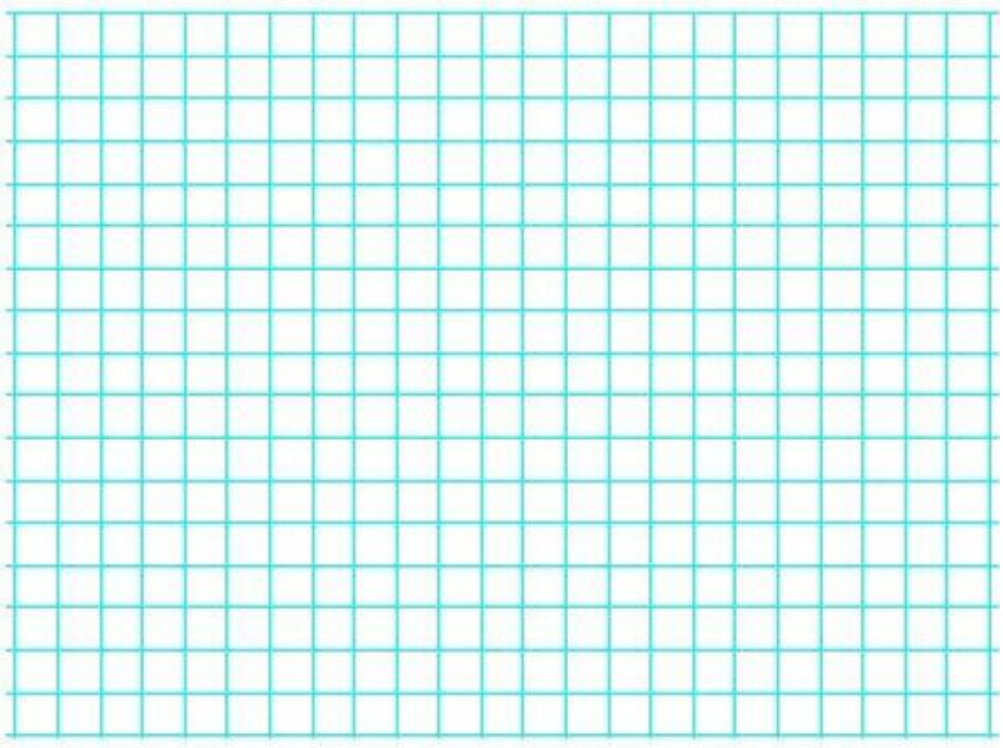
1) How many concurrent threads can the system run: _____ _. This value is now referred to as the max_num_threads.

2) How many microseconds does it take a single thread to run 10,000 calculations: _____

3) How many microseconds does it take 2 threads to run 10,000 calculations: _____. Assume the number of calculations is evenly distributed among the two threads.

4) Why is the time from #3 not half the time from #2?

5) Starting with one thread and then working up to the **max_num_threads** by adding one more thread at a time create a plot below with Number of Threads vs Processing Time (microseconds) for 1 billion calculations.



6) How many microseconds does it take 1,000 threads to run 1,000 calculations: _____. Assume the number of calculations is evenly distributed among the threads.

7) Assuming that the processing time for a single calculation is negligible and using the results from Question #6, how long does it take to create a single thread in microseconds? _____

APPENDIX A: ECE2036 Lab Grading Rubric

If a student's program runs correctly and produces the desired output, the student has the potential to get a 100 on his or her lab; however, TA's will **randomly** look through this set of "perfect-output" programs to look for other elements of meeting the lab requirements. The table below shows typical deductions that could occur.

In addition, if a student's code does not compile, then he or she will have an automatic 30% deduction on the lab. Code that compiles, but does not match the sample output can incur additional deductions from 10% to 30% depending on how poorly the output matches the output specified by the lab. This is in addition to the other deductions listed below or due to the student not attempting the entire assignment.

AUTOMATIC GRADING POINT DEDUCTIONS

Element	Percentage Deduction	Details
Does Not Compile	30%	Programs do not compile on PACE-ICE!
Does Not Match Output	10%-30%	The programs compile but don't match all output
Answers on Turn-In Sheet	40%	Make sure you fill the turn-in sheet and post this to canvas.

ADDITIONAL GRADING POINT DEDUCTIONS FOR RANDOMLY SELECTED PROGRAMS

Element	Percentage Deduction	Details
Turn in Sheet Answers	30%	Incorrect answers on turn in sheet
Multi-Threading	40%	Threading not implemented correctly
Pi Calculation	10%	Value of pi is not correctly calculated
Clear Self-Documenting Coding Styles	5%-20%	This can include incorrect indentation, using unclear variable names, unclear comments, or compiling with warnings.

LATE POLICY

Element	Percentage Deduction	Details
Late Deduction Function	score - (20/24)*H	H = number of hours (ceiling function) passed deadline note : Sat/Sun count has one day; therefore H = $0.5*H_{weekend}$