Lab 4

This homework has four parts. In each you can change the code I provide to work better with what you write, including fixing possible bugs. Your code should be written to be thread safe. Use the Scholar queue to do your timing runs.

Part 1:

Using the code in Lab4Loop.c, modify the main procedure that has a parallel for loop that calls doWork on each element of wA. Comments in the program will say what scheduling to use. Think about the different times and why there are different times.

Part 2:

For the scheduling strategy that works best for Part 1, quadruple the number of threads that execute the loop. Does the loop run faster or slower?

Part 3

Using the code in LabSection.c, write a parallel version that uses OpenMP parallel sections. Use 4 sections, where T is the number of threads supported by your hardware. Each section should have a loop that is pulling work from the wA array. The work in each element of wA should be performed once, and only once. Compare the time with the sequential time for the loop.

Part 4 (If you have time, and this one is harder):

Using the code in Lab4Parallel.c, write a main routine that initializes and adds work to the work queue, and then add a loop within a parallel construct that pulls work from the work queue. Compare the execution time with the sequential execution time.