**Operating Systems**

**Mini-homework**

This homework extends the in-class exercise. There are 2 pages of instructions.

Add the following functionality **exactly** as specified, to allow your program to be tested automatically.

The arguments to the program (let’s call it mm) must be:

mm rows cols val1 val2

These arguments mean that you should generate a matrix **m1**:

Size: rows X cols

Every cell contains the value val1

And a matrix **m2**:

Size: cols X rows

Every cell contains the value val2

The maximum number of rows or cols your program should handle is 1024.

Your program should multiply the two matrices (**m1 X m2**) using a sequential method and print out the elapsed time to stderr.

Your program should then multiply the two matrices using one thread per row (in m1) and print out the elapsed time to stderr.

Your program should print out the result matrix of **only** the parallel computation to stdout using exactly the code below (which prints to stdout with printf). Of course, you will use your own variable names, but the output spacing must be **exactly** as written here.

for ( i = 0; i < resrows; i++ )

{

for ( j = 0; j < rescols; j++ )

printf( " %d ", result\_T[i][j] );

printf( "\n" );

}

Your program should print absolutely nothing else to stdout; any debugging statements you have for yourself should go to stderr.

You may use global variables freely.

**Submission Requirements**

Submit exactly one C file and one document (txt, docx, pdf) containing the answers to the following questions. Do not zip these files up, and do not include any spaces in your filenames.

1. What are the processor and memory specifications of the machine you are running your program on?
2. List the real time in microseconds that your program took with the following inputs (on the second try):

mm 10 20 2 4

mm 50 70 2 4

mm 100 200 2 4

mm 300 500 2 4

mm 500 700 2 4

mm 900 1000 2 4

mm 1023 1023 2 4

1. What is the pattern you see in the time difference between the sequential and parallel implementations, and how do you explain your results?
2. Did your threaded solution require any synchronization mechanisms? Explain why or why not.