TCSS 422 — Computer Operating Systems
Winder 2015 — Homework Assignment 2

Due Date: Saturday, Jan. 24

Guidelines

Homework should be electronically submitted to the instructor by the end of the day on the due date. A submission link is provided on the course Canvas page for this assignment.

Assignment Description

This assignment is intended to introduce you to the Pthreads library and give you a little more experience with C. In this assignment you will implement a small multithreaded program for a classic parallel programming algorithm: matrix product, a.k.a. matrix multiplication. Matrix products are an extremely common and important numerical operation used in a wide variety of computer science applications, such as 3D graphics, artificial intelligence, data compression, and cryptography. In this assignment you'll focus on just computing a matrix product without any specific application. Starter code for this assignment is available on the course Canvas page for this assignment.

Implementation Specifications

The provided `main.c` source file contains a skeleton for the matrix product function you're to implement, `multithreaded_matrix_product`, as well as an example of its invocation in the `main` function. You will need to implement your own testing mechanisms.

While matrices are conceptually two-dimensional structures, within this program they are stored as one-dimensional arrays in row-major order. For example, imagine a 6-by-4 matrix (6 rows, 4 columns) as shown below. This matrix is stored in an array of 24 elements. The first four elements of the array hold the first row of matrix elements, the second four elements hold the second row of matrix elements, and so on.

A 6-by-4 matrix:

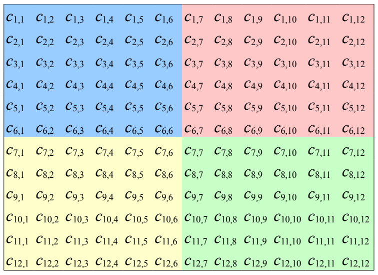
0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15
16	17	18	19
20	21	22	23

The matrix's array representation:

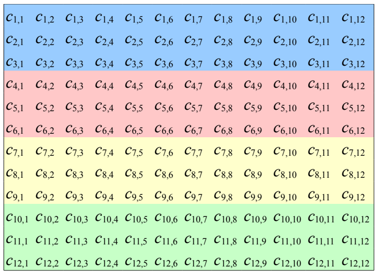
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23

```
int * multithreaded_matrix_product(const int A[], const int B[],
                                   int r, int s, int t)
```

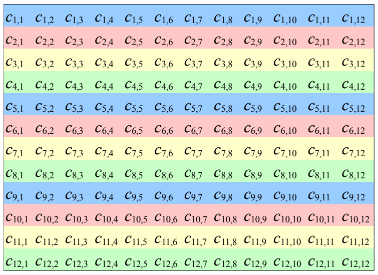
This function should return a new, dynamically allocated matrix that is the product of matrices A and B (in that order, remember that the matrix product is not commutative). The product will be referred to as matrix C hereafter. A is an r-by-s matrix and B is an s-by-t matrix, thus C will be an r-by-t matrix. Calculation of the product should utilize four threads, theoretically improving the speed of the multiplication by up to a factor of four. Each element of matrix C is the dot product of a row of matrix A and a column of matrix B. The traditional way to split this work among multiple threads is to partition the elements of C and make each thread responsible for computing a different subset of the partition. For example, using four threads the most straightforward ways to partition the elements of C are by quadrant (a) or by row, either in blocks (b) or interleaved (c). When the width and/or height of C is not evenly divisible some threads may do slightly more work than others.



(a)



(b)



(c)

One of the major advantages of these forms of partitioning is that there is no problematic data contention. While multiple threads will read the same data from matrices A and B, each element of matrix C is written by only one thread and there are no race conditions. Consequently, no thread synchronization is needed. Calculation of the matrix product is complete when all four threads terminate.

<u>Deliverables</u>

The following items should be submitted to the instructor. Failure to correctly submit assignment files will result in a 10% grade penalty.

1) The completed main.c source file.
2) Any new supporting files, e.g., additional source code, needed by main.c.

Do not include any extraneous files, such as Eclipse IDE files, object files, or subversion files.