Matrix Multiplication Project

Given two matrices, A and B, where matrix A contains M rows and P columns and matrix B contains P rows and N columns, the matrix product of A and B is matrix C, where C contains M rows and N columns. The entry in matrix C for row i, column j (Ci, j) is the sum of products of the elements for row i in matrix A and columns j in matrix B. That is,

p

Ci, j =Σ Ai, k x Bk, j

k=1

For example, if A is a 3-by-2 matrix and B is a 2-by-3 matrix, element C3, 1 is the sum of A3, 1 x B1, 1 and A3, 2 x B2, 1

C3, 1 = A3, 1 x B1, 1 + A3, 2 x B2, 1

For this project, calculate each element Ci, j in a separate *worker* thread. This will involve creating M x N worker threads. The main – or parent – thread will initialize the matrices A and B and allocate sufficient memory, for matrix C which will hold the product of the matrices A and B. These matrices will be declared as global data so that each worker thread has access to A, B and C. The matrices A and B can be populated by reading in values from a file.

Passing Parameters to Each Thread

The parent thread will create M x N worker threads, passing each worker the values of row i and column j that it is to use in calculating the matrix product. This requires passing two parameters to each thread. The easiest approach with pthreads is to create a data structure using a struct. The members of this structure are i and j, and the structure appears as follows:

/\* Structure for passing data to threads \*/

typedef struct {

int i; /\* Row \*/

int j; /\* Column \*/

} Params;

The pthreads program will create worker threads using a strategy similar to that shown below:

/\* Create M x N worker threads \*/

for (int i = 0; i < M; i++) {

for (int j = 0; j < N; j++) {

Params \*data = (Params \*) malloc (sizeof (Params));

data->i = i.

data->j = j;

/\* Now create the thread passing it data as a parameter \*/

}

}

Input

You are given two matrices A and B of type float.

float A[m][k], B[k][n];

Where:

#define m 20

#define k 20

#define n 20

Result matrix:

float C[m][n];

The matrices are statically initialized. You should embed them in your program. The program should print the result matrix and the output should be redirected to a text file, as in:

* ./matMult > resultMat.txt

What to submit on Blackboard

1. Header file (.h), if any and the C program (.c) file. Please do not submit binaries
2. The output matrix file.
3. Any documentation.