CSCI 176 (Parallel Processing) Spring 2021 Program 3 (20 pts) Due: March 19 (F)

Write a Pthreads parallel program for matrix multiplication.
The matrix multiplication problem is defined as:

A(L*m) * B(m*n) = C(L*n)

where, A, B and C are L*m, m*n and L*n matrices, respectively, and $C_{i,j} = \sum_{k=0}^{m-1} (A_{i,k} * B_{j,k})$

Your program should access the command line arguments for the matrix dimensions, i.e., L, m and n, together with the number of threads used for the computation.
The matrices should be declared as global (shared among all threads), and your main process initializes the values with the following formula: $A_{i,j} = i+j+1$ and $B_{i,j} = i+j$
For this, you need to declare the matrices as dynamic 2-dimensional arrays.

After initializing A and B matrices, the main process should create the requested number of threads and let them work for building C matrix, i.e., entries of the C matrix are computed and updated by those slave threads.
To distribute the task of completing the C matrix to those multiple threads fairly, please assign rows (total number of rows = L) of the C matrix to the threads in a cyclic way (jump-p way), i.e., for example with P=4, Th_0 is assigned with row_0, Th_1 is assigned with row_1, Th_2 is assigned with row_2, Th_3 is assigned with row3, then, Th_0 is assigned with row_4, Th_1 is assigned with row_5, and so on.

Check the correctness of your program with small size matrices first; and then use the following dimensions of the matrices for the final output to submit:
L=1003, m=2000, n=3000
Since the size of the output is extremely big (~36MB), please display only [first_10 * first_10] and [last_10 * last_10] entries of matrix C; this should be done in the main process after joining the threads.

Run your program with number of threads = 1, 2 , 4, 8 and check/show the execution time for each run. For this, you need to use time checking codes in your program.

* Finally, please try the B^T way of optimization, i.e., A(L*m) * B^T(n*m) = C(L*n), and check how it affects the execution time.

Submission:

Include good documentations and submit the hard copy of the source code and the run time output (from 4 runs). I suggest you use the "typescript" way, but it is not required.

Sample output (run with number of threads = 4):

```
L=1003, m=2000, n=3000
Thread_2:  2 ~ 1002, step 4
Thread_0:  0 ~ 1002, step 4
Thread_1:  1 ~ 1002, step 4
Thread_3:  3 ~ 1002, step 4
=== C:  [first_10 * first_10] ===
. . . .
. . . .
=== C: [last_10 * last_10] ===
. . . .
. . . .
Time taken (sec) = 33.079007
```