CSE344 Homework5 Report

Yeşim Yalçın-200104004094

1-) My Solution Approach

Command Line Argument Analysis
My first action was to check if the command line arguments were inputed properly.
The data file names can be relative or absolute. n must be bigger than 2, m must be at least 2.

Reading the Data Files
Data files must exist and include at least (2^n*2^n) characters. If it inclues less, the program exits. If it includes more, they are ignored.
Reading is done character by character. The read characters are converted into their corresponding integer according to ASCII table. The data read from the first file is put inside matrixA. The data read from the second file is put inside matrixB. They are held in global so that threads can reach them easily. matrixA and matrixB will not be changed after first set. Therefore it is not necessary to protect them with mutex in threads.
NOTE: Because the reading is done character by charecter if the data file is too long, it might take a few seconds to read both files. Example of it for a 256*256 file is included in "Acceleration Tests" part. If you see nothing happening while trying huge data, it is not busy waiting or deadlocking. It is processing. You can send SIGINT to check in which part of the processing the main process is. Because I included SIGINT received outputs.

Threads
After the data is read into the memory, allocation for result matrixes is done. matrixC is to hold the multiplication result of the matrixA and matrixB. transformResult struct is to hold the imaginer and reel parts of the result of 2D DFT. They are also held in global so that threads can reach these datas easily. Each thread will have access to read all elements inside these arrays however, they will modify only specific elements. No two thread is able to modify the same element. Therefore, it is not needed to protect them with a mutex either.
After the allocation, m threads are created.
While creating the threads, they are given a thread number and according to their thread number, the columns they will be modifying will be calculated as well. These information is stored inside a struct and given as an argument to thread functions.
Threads start by calculating matrixC. When a thread finishes first part, it increments a global variable named "arrived". Then it waits till all other threads also finish first part and arrived becomes equal to m count. This synchronization barrier is done via 1 mutex and 1 condition variable. When the last thread finishes first part, it broadcasts and wakes up all other threads. Then all of them start the second part.
In the second part, 2D DFT is calculated according to the formula:

$$F(x,y) = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m,n)e^{-j2\pi(x\frac{m}{M}+y\frac{n}{N})}$$

Here, M-1 and N-1 are size. m, n, x, y are for iterating in loops. (There are 4 for loops inside eachother). e here is converted into corresponding sin and cos functions.
From this formula, reel and imaginer parts of the result is calculated.
After all threads finish calculation, the main process returns from joining all threads and writes the result to the given output file. Output file does not have to exist from before.

SIGINT
When SIGINT is sent to the process, a random thread catches it. I have a sigint handler which sets a sigintFlag as 1. If any process sees this flag as 1, it broadcasts (if necessary) to wake up other threads and the threads which see this, returns. The main process wakes up from join and exits.

2-) Acceleration Tests

Testing

- **n=8, m=2 Test**

May 18 +03 2022 13 30 32: Two matrices of size 256x256 have been read. The number of threads is 2.
May 18 +03 2022 13 31 29: The process has written the output file. The total time spent is 56.900079 seconds.

- **n=8, m=4 Test**

May 18 +03 2022 13 52 09: Two matrices of size 256x256 have been read. The number of threads is 4.
May 18 +03 2022 13 52 38: The process has written the output file. The total time spent is 28.829230 seconds.

- **n=8, m=8**

May 18 +03 2022 13 55 16: Two matrices of size 256x256 have been read. The number of threads is 8.
May 18 +03 2022 13 55 34: The process has written the output file. The total time spent is 17.561746 seconds.

- **n=8, m=16**

May 18 +03 2022 13 56 44: Two matrices of size 256x256 have been read. The number of threads is 16.
May 18 +03 2022 13 57 00: The process has written the output file. The total time spent is 16.204375 seconds.

Observation
I have 8 cores in my CPU. Here we can see that when the thread numbers are doubled, the total time were almost halved until 8 threads. There was not a significant change between 8 and 16 threads because of the core count.

Testing

- **n=4, m=4 Test**

May 18 +03 2022 13 59 16: Two matrices of size 16x16 have been read. The number of threads is 4
May 18 +03 2022 13 59 16: The process has written the output file. The total time spent is 0.002177 seconds.

- **n=6, m=4 Test**

May 18 +03 2022 13 59 46: Two matrices of size 64x64 have been read. The number of threads is 4
May 18 +03 2022 13 59 46: The process has written the output file. The total time spent is 0.117350 seconds.

- **n=8, m=4 Test**

May 18 +03 2022 14 00 22: Two matrices of size 256x256 have been read. The number of threads is 4
May 18 +03 2022 14 00 51: The process has written the output file. The total time spent is 28.774338 seconds.

Observation
When n increases, the total time increases exponentially (or even faster).

3-) The Parts I Have Completed

I have completed all the requirements of the homework and tested in various ways.