Machine Problem 4

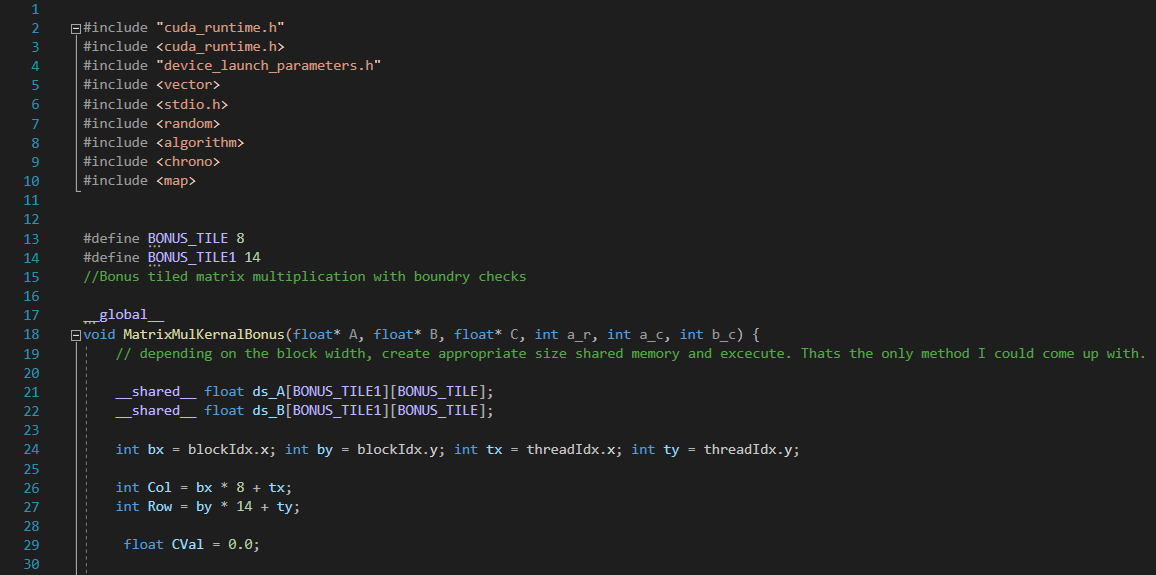
Name: Christopher Masloub

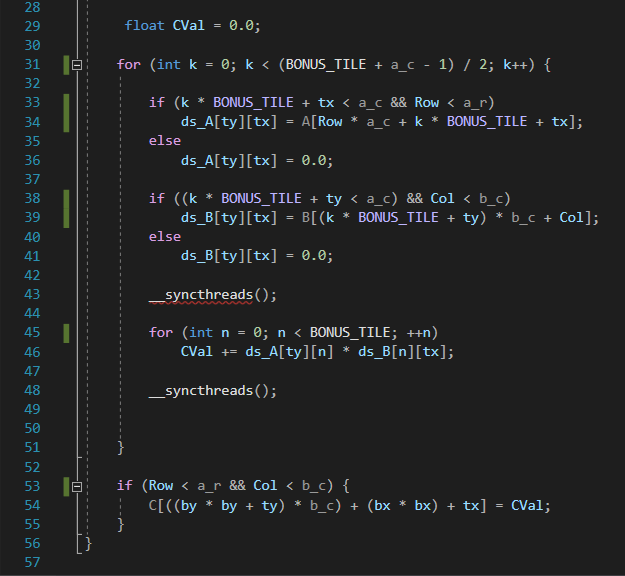
Student Number: 20052223

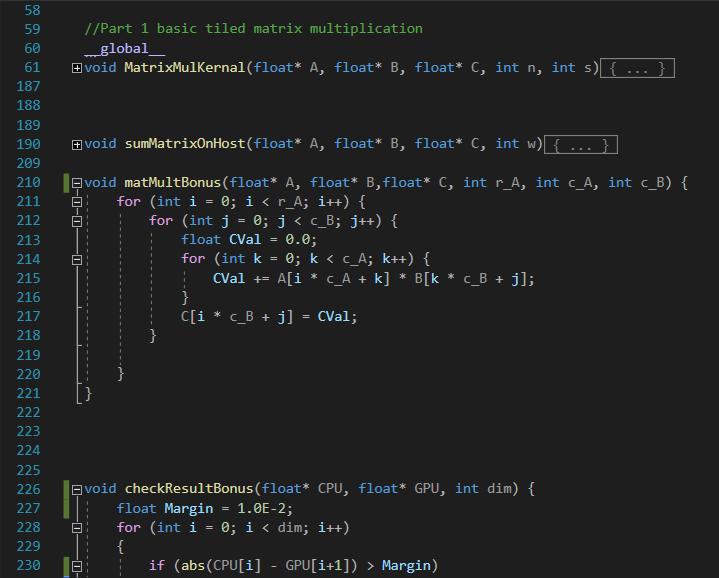
DISCLAIMER: This report was done on my own laptop using a gtx 1650 mobile due to the fact that the computation was taking too long on the remote PC especially since I had to run the CPU portion multiple times to get good median values. On the remote PC with my slow internet, the operation would often stop due to the congestion from everyone using it. My code is included in the uploaded Kernal.cu and my outputs are in a text file in the submission folder in order to make the report cleaner. In addition this is indeed MP4 and not MP3-copy.

# Part 1:

## Code:

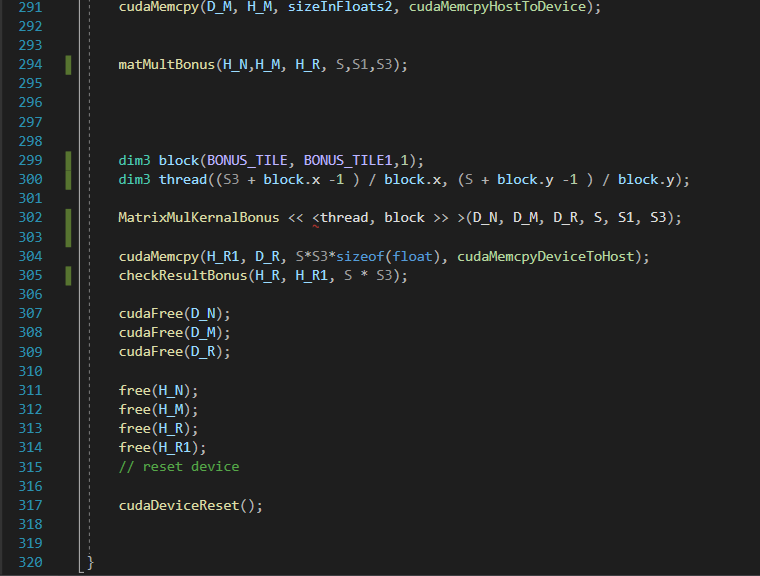


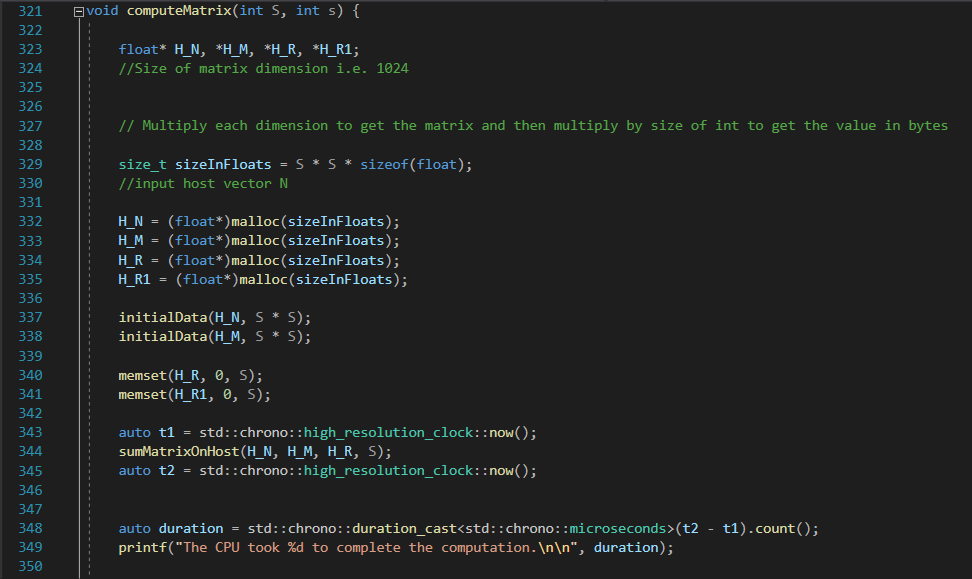


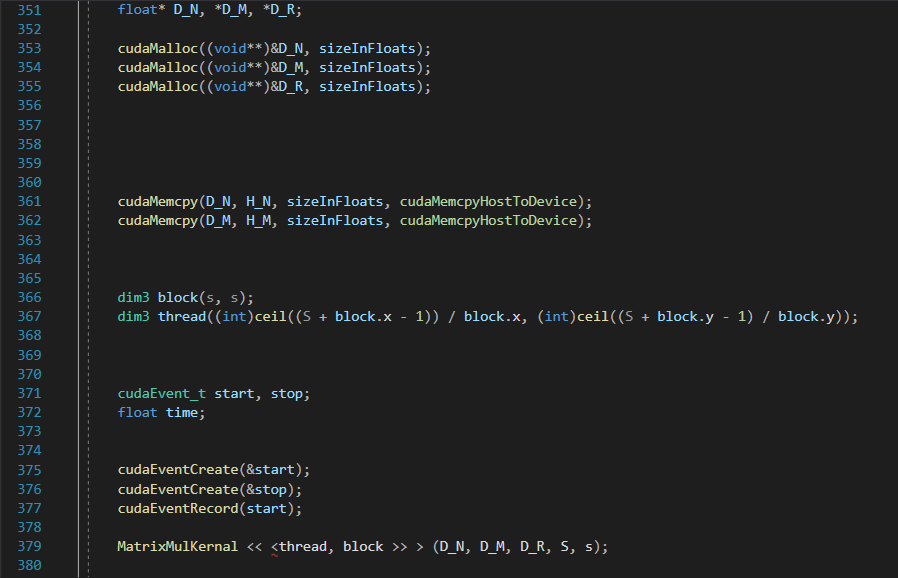


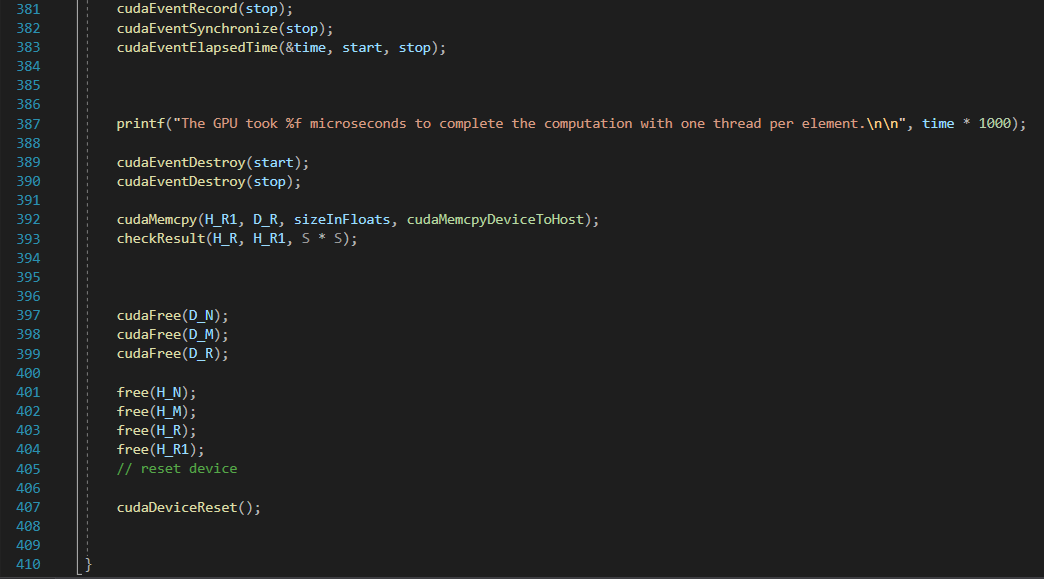








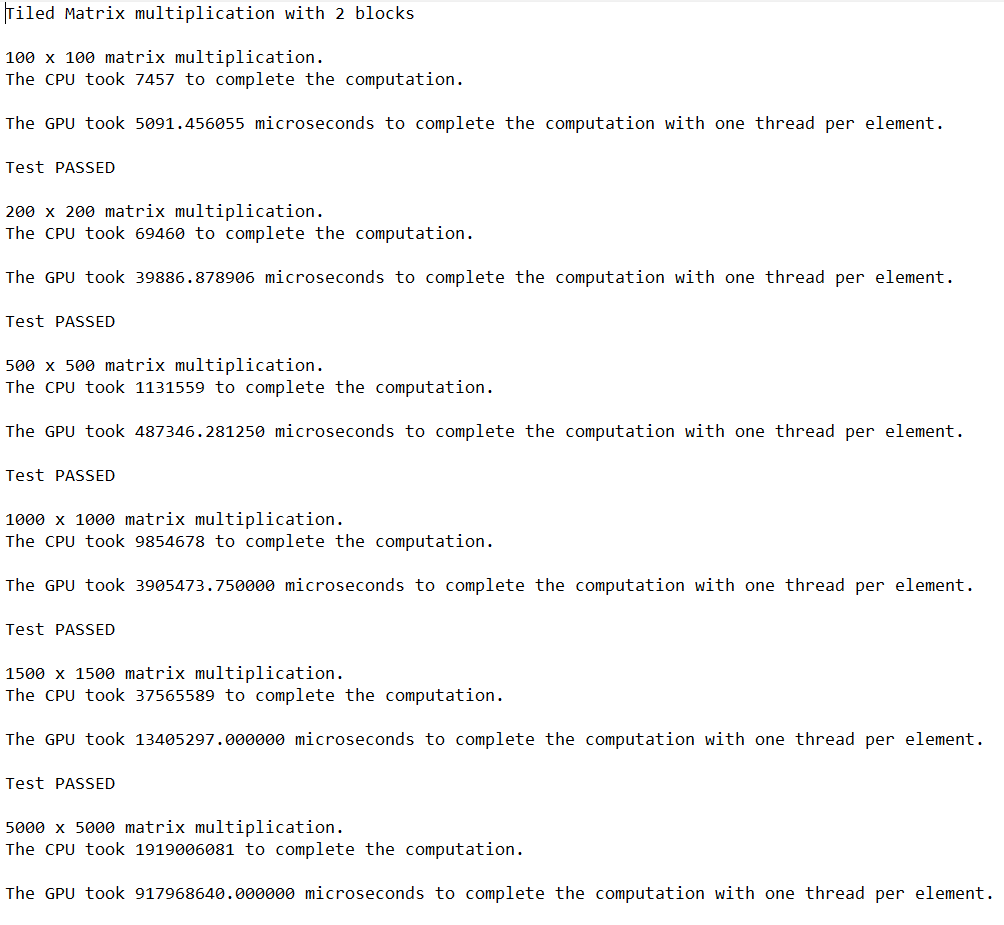


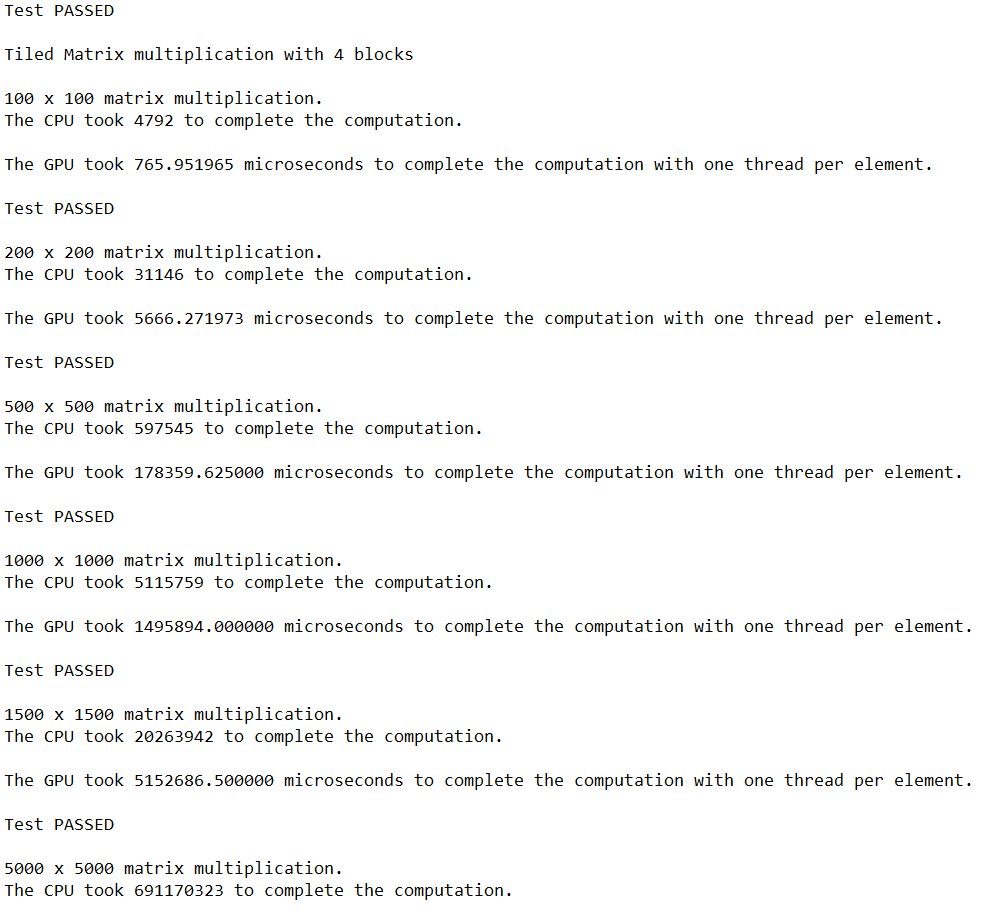


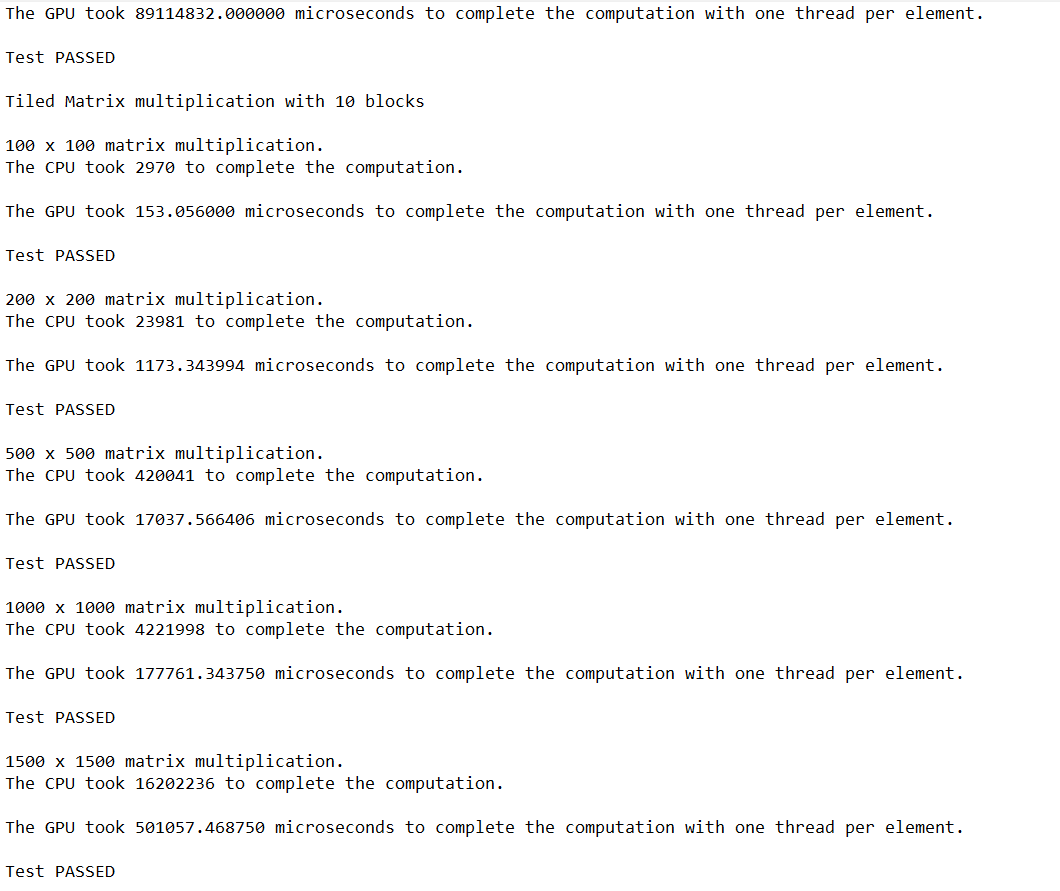


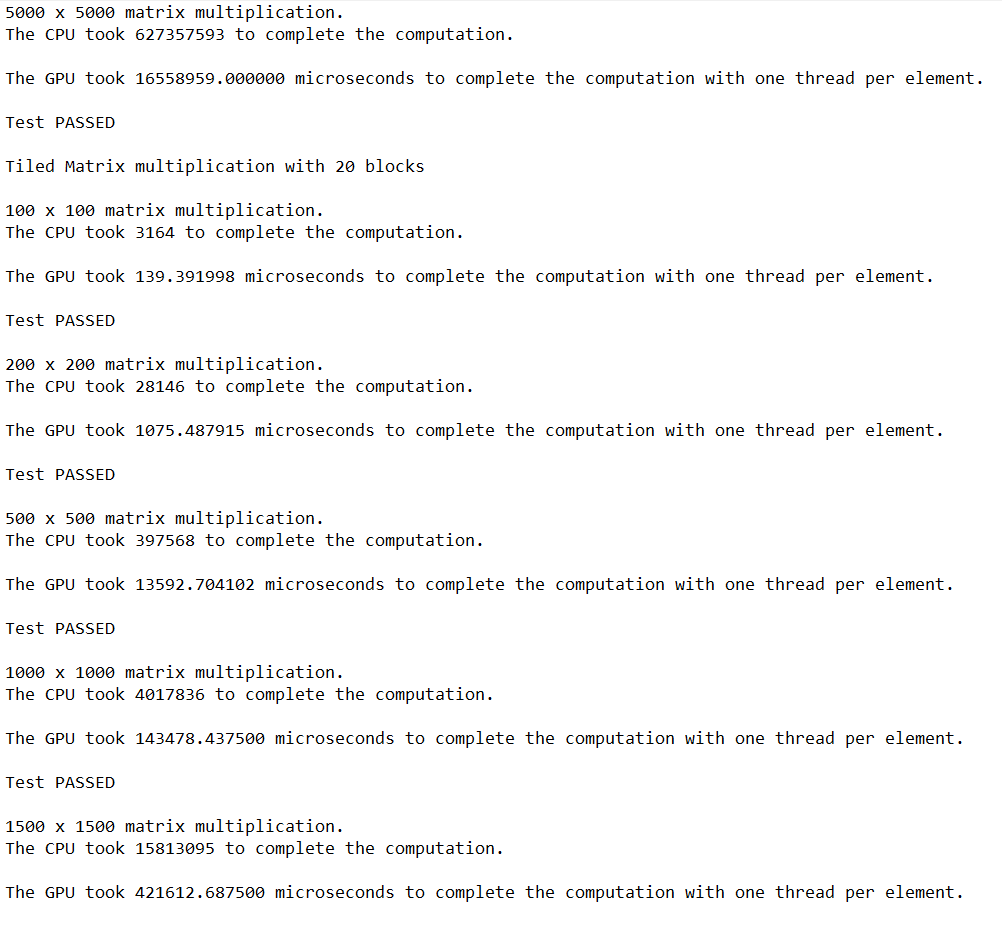


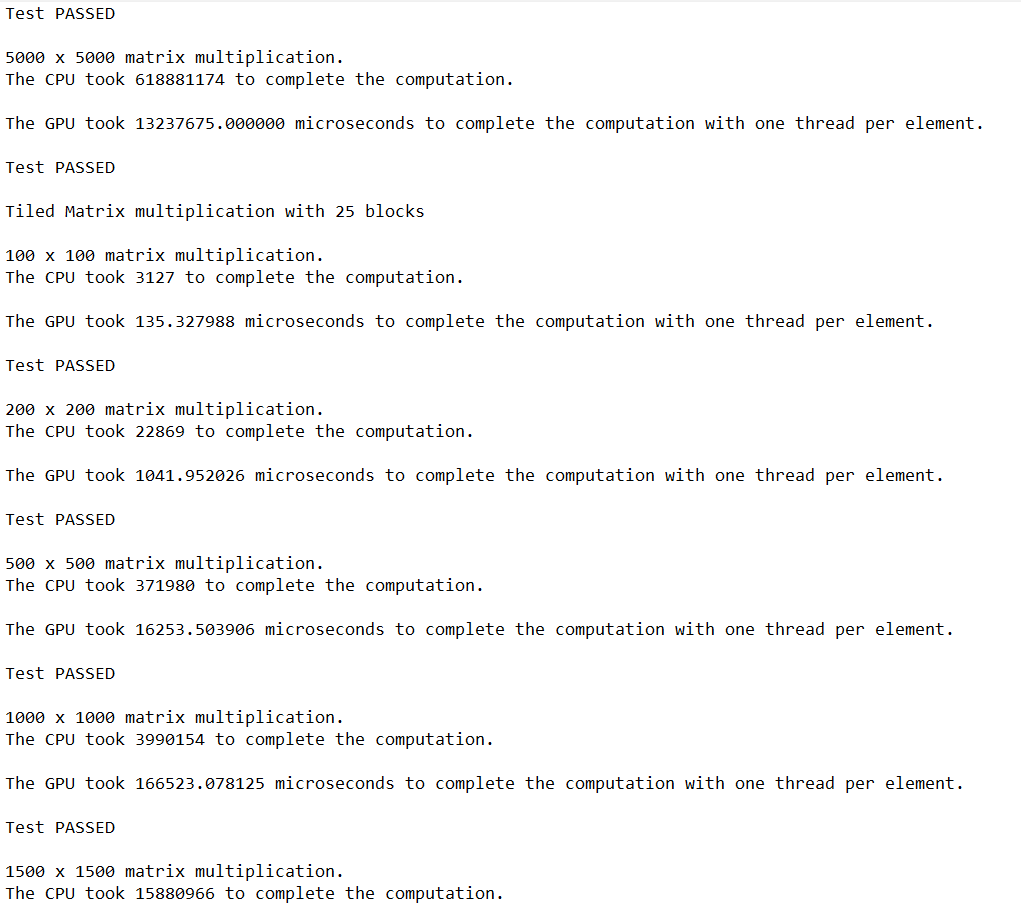
## Output:

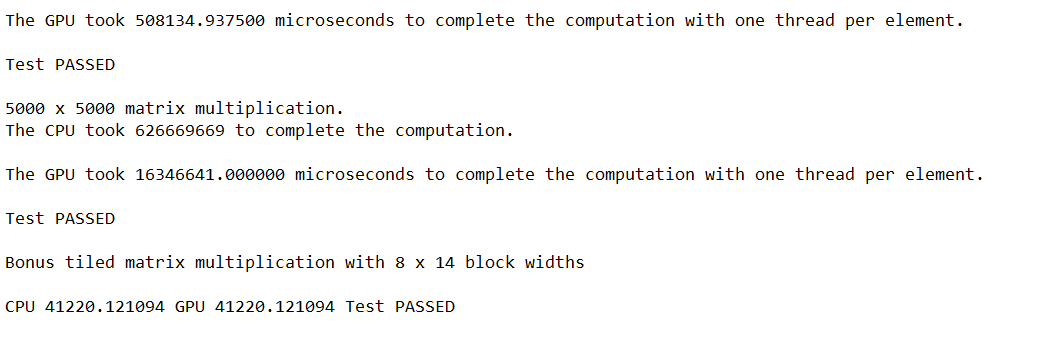












## Data:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Time (microseconds) | | | | | |
| Matrix Size | CPU | Tile Width 2 | Tile Width 4 | Tile Width 10 | Tile Width 20 | Tile Width 25 |
| 100x100 | 4792 | 5091.456055 | 765.951965 | 153.056 | 139.391998 | 135.327988 |
| 200x200 | 31146 | 39886.878906 | 5666.271973 | 1173.343994 | 1075.487915 | 1041.952026 |
| 500x500 | 597545 | 487346.28125 | 178359.625 | 17037.566406 | 13592.704102 | 16253.503906 |
| 1000x1000 | 5115759 | 3905473.75 | 1495894 | 177761.34375 | 143478.4375 | 166523.078125 |
| 1500x1500 | 20263942 | 13405297 | 5152686.5 | 501057.46875 | 421612.6875 | 508134.9375 |
| 5000x5000 | 691170323 | 917968640 | 89114832 | 16558959 | 13237675 | 16346641 |

## Analysis:

It is evident from the results demonstrated in the following graph that using tiled matrix multiplication was the fastest with 25, 20 and 10 blocks, even 4 blocks was exponentially faster than the baseline matrix multiplication using the CPU. The only odd result was that using a block width of 2 resulted in the CPU being faster. Otherwise, tiled matrix multiplication with 4 blocks and above is exponentially faster.

The following are done based on the remote PC

1. The number of threads that can be scheduled with 14 SMs is 28672 threads.
   1. The number of registers per thread will be 63.
   2. The shared memory Size is 64 KB.
   3. Blocks per SM is TILEWIDTH\*TILEWIDTH/14.
   4. Maximum total threads simultaneously scheduled are 3584 using the following knowledge with 14 SMs, 8 CUDA cores and 32 threads per core.

|  |  |  |  |
| --- | --- | --- | --- |
| Number of Registers | Shared Memory Size | Number of Blocks per Streaming Multiprocessor | Maximum Total Threads Simultaneously Scheduled |
| 63 per thread | 64KB | Tile width x tile width / 14 | 14 SMs  8 CUDA cores  32 threads per core  =3584 |

# Bonus:

BONUS: The bonus Tiled Matrix Multiplication with boundary checks for different matrix dimensions as well as different Tile Widths was implemented as well. The only issue was implementing the CPU function that does the same thing, the CPU version is not as accurate and needs to be adjusted. The code for both as well as their instantiation is available in the kernel.cu file in the submission folder.