ASSIGNMENT - 1

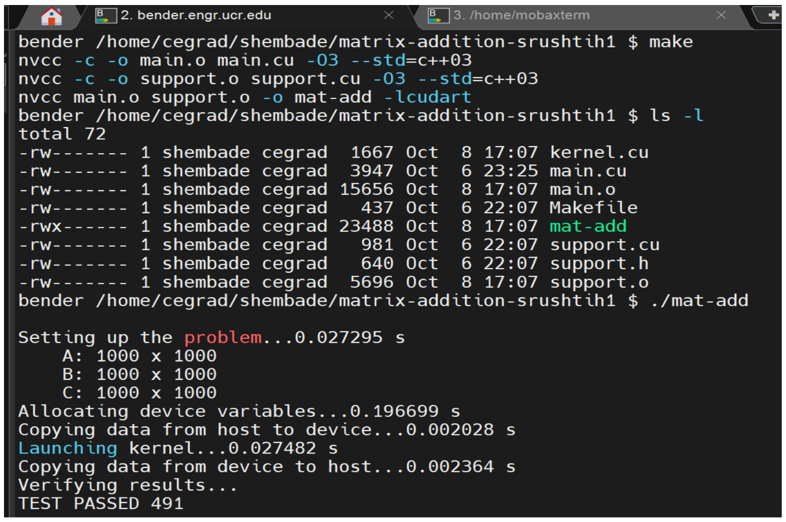
Name - Srushti Hembade
Student Id - 862395839
Net id - shemb001

Question 1: How many total thread blocks do we use?
→ As the given matrix size in the question to be 1000*1000, and the default thread block size as 16*16=256, we calculate the total thread blocks as below -
(1000*1000)/256 = 3906.25 blocks
We need the count of thread blocks in integer, so we can consider the integer value as 3907 thread blocks in this case.

Question 2: Are all thread blocks full? That is, do all threads in the thread block have data to operate on?
→ No, not all thread blocks are full as the matrix size of the vector we trying to add is not equal to the block size. So there cannot be a case where there will be data present to operate on for all thread blocks.

Question 3: How can this basic Matrix Addition program be improved? (What changes do you think can be made to speed up the code?)
→ If the shared memory and constant memory can be used to store data and run programs, matrix addition program speed should rise as it will reduce the global memory access. Also, using smaller blocks could help, as it in turn eases memory contention.

OUTPUT screenshot for the Matrix Addition -



```
bender /home/cegrad/shembade/matrix-addition-srushtih1 $ make
nvcc -c -o main.o main.cu -O3 --std=c++03
nvcc -c -o support.o support.cu -O3 --std=c++03
nvcc main.o support.o -o mat-add -lcudart
bender /home/cegrad/shembade/matrix-addition-srushtih1 $ ls -l
total 72
-rw------- 1 shembade cegrad  1667 Oct  8 17:07 kernel.cu
-rw------- 1 shembade cegrad  3947 Oct  6 23:25 main.cu
-rw------- 1 shembade cegrad 15656 Oct  8 17:07 main.o
-rw------- 1 shembade cegrad   437 Oct  6 22:07 Makefile
-rwx------ 1 shembade cegrad 23488 Oct  8 17:07 mat-add
-rw------- 1 shembade cegrad   981 Oct  6 22:07 support.cu
-rw------- 1 shembade cegrad   640 Oct  6 22:07 support.h
-rw------- 1 shembade cegrad  5696 Oct  8 17:07 support.o
bender /home/cegrad/shembade/matrix-addition-srushtih1 $ ./mat-add

Setting up the problem...0.027295 s
    A: 1000 x 1000
    B: 1000 x 1000
    C: 1000 x 1000
Allocating device variables...0.196699 s
Copying data from host to device...0.002028 s
Launching kernel...0.027482 s
Copying data from device to host...0.002364 s
Verifying results...
TEST PASSED 491
```