Simple Matrix Multiplication
(Download ZipFile)

1) Unzip lab1-simple-matrixmul into your sdk projects directory

2) Edit the MatrixMulOnDevice(...) function in matrixmul.cu and the MatrixMulKernel(...) function in matrixmul_kernel.cu to complete the functionality of the matrix multiplication on the device. Do not change the source code elsewhere. The size of the matrix is defined such that one thread block will be sufficient to compute the entire solution matrix.

3) There are several modes of operation for the application.

No arguments: The application will create two randomly initialized matrices to multiply. After the device multiplication is invoked, it will compute the correct solution matrix using the CPU, and compare that solution with the device-computed solution. If it matches (within a certain tolerance), if will print out "Test PASSED" to the screen before exiting.

One argument: The application will use the random initialization to create the input matrices, and write the device-computed output to the file specified by the argument.

Two arguments: The application will initialize the two input matrices with the values found in the files provided as arguments. No output is written to file.

Three arguments: The application will read its inputs from the files provided by the first two arguments, and write its output to the file provided in the third.

Note that if you wish to use the output of one run of the application as an input, you must delete the first line in the output file, which displays the accuracy of the values within the file. The value is not relevant for this application.

4) Answer the following questions:

1. How many times is each element of the input matrices loaded during the execution of the kernel?

2. What is the memory-access to floating-point computation ratio in each thread? Consider a multiply and addition as separate operations, and ignore the storing of the result. Only global memory loads should be counted towards your off-chip bandwidth

Grading:

Your submission will be graded on the following parameters.

Demo/knowledge: 25%

- Produces correct result output file for our test inputs.

Functionality: 40%

- Correct usage of CUDA library calls and C extensions.
- Correct usage of thread id's in matrix computation.

Report: 35%

- Answer to question 1: 15%, answer to question 2: 20%