OpenCL exercise 4: Matrix Multiplication

1 Introduction

The task in this exercise is to implement matrix multiplication on the GPU.
Let **A**, **B**, **C** be 3 matrices. The matrix product $C = AB$ is defined by

$$C_{i,j} = \sum_{k=1}^{n} A_{i,k} B_{k,j}$$

In this exercise we use sizes which are an integral multiple of the work group size.

2 Syntax: Local memory

```
__local int i; // Declare "i" as 32-bit integer in local memory
__local uint a[10][15]; // Declare "a" as 2D array of unsigned 32-bit integers with a size of 10x15

barrier(CLK_LOCAL_MEM_FENCE); // Wait until all threads have reached this point and prevent
                              // any local memory accesses from being moved accros this line
```

3 Task 1: Simple GPU implementation

Write the kernel `matrixMulKernel1` using one work item for calculating each element of the matrix C. Check that the results match the results from the CPU. Add the code needed for measuring the speedup.

4 Task 2: Local memory

Local memory allows the threads inside of a work group to cooperate and is much faster to access than global memory.
Implement another version of the matrix multiplication in which the matrices are split into submatrices where each submatrix has the size of a work group. The data for two submatrices (one from A and one from B) are loaded into the local memory, a barrier makes sure that all threads have reached that point, the product of the submatrices is computed and added to the result of the multiplication of others submatrices and another barrier is executed.
Write the kernel `matrixMulKernel2` to implement the matrix multiplication as stated above. Make sure that the results are the same as on the CPU. Compare the performance with the first matrix multiplication.

5 Optional: Task 3

Make a copy of the kernel of Task 2 and modify it so that it doesn't need any compile-time knowledge of the work group size (i.e. `WG_SIZE`).

- Will need a dynamically sized local memory area
- Will be slower than kernel 2

6 Optional: Task 4

Create a kernel similar to kernel 1, but use OpenCL images for A and B

- Will be slower than kernel 2 and kernel 3
- Might be slower or faster than kernel 1