OpenCL exercise 5: Prefix sum

The aim of this exercise is to calculate the prefix sum for an array on the GPU.
See `http://en.wikipedia.org/wiki/Prefix_sum`
The prefix sum is not easy to parallelize, because different work items have to communicate with each other. Inside a work group this is possible using `barrier()` and local memory. Between work groups no synchronization is possible. Because we have to wait for other work groups to finish, we use several kernel launches: The second kernel will only start after all work items in first kernel have finished.

The GPU implementation should contain 5 kernel launches:

1. Launch the prefix sum kernel with 256 * 256 work groups, each with 256 work items. In addition to the normal results, each work group will write one sum into `d_temp1` in global memory.
2. Launch the prefix sum kernel with 256 work groups, each with 256 work items. In addition to the normal results, each work group will write one sum into `d_temp2` in global memory.
3. Launch the prefix sum kernel with 1 work group, with 256 work items. No overall sum is written into global memory.
4. Launch the block add kernel with 256 work groups, each with 256 work items.
5. Launch the block add kernel with 256 * 256 work groups, each with 256 work items.

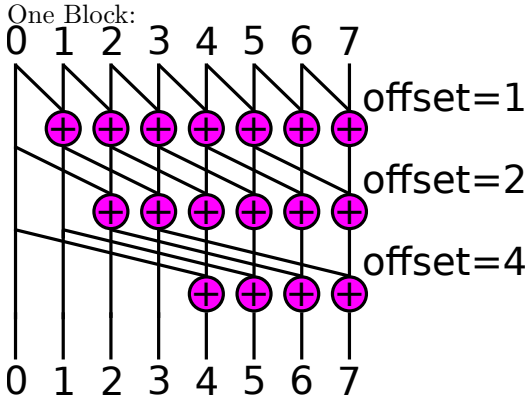
The prefix sum kernel itself should:

1. Load input data to local memory
2. Loop over offsets
3. Write results to global memory

The block add kernel should for each block add one value to all the elements of the block.

Add the usual code for measuring the performance on the CPU and on the GPU with/without memory transfers, including the number of items processed per second.

One Block:



All Blocks: