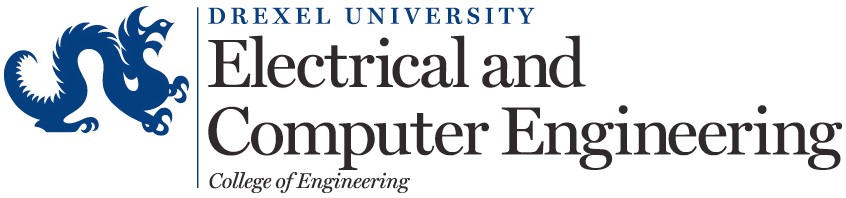
****

Drexel University

Electrical and Computer Engineering Dept.

ECEC-413

**Counting Sort with CUDA**

**Chris Kasper**

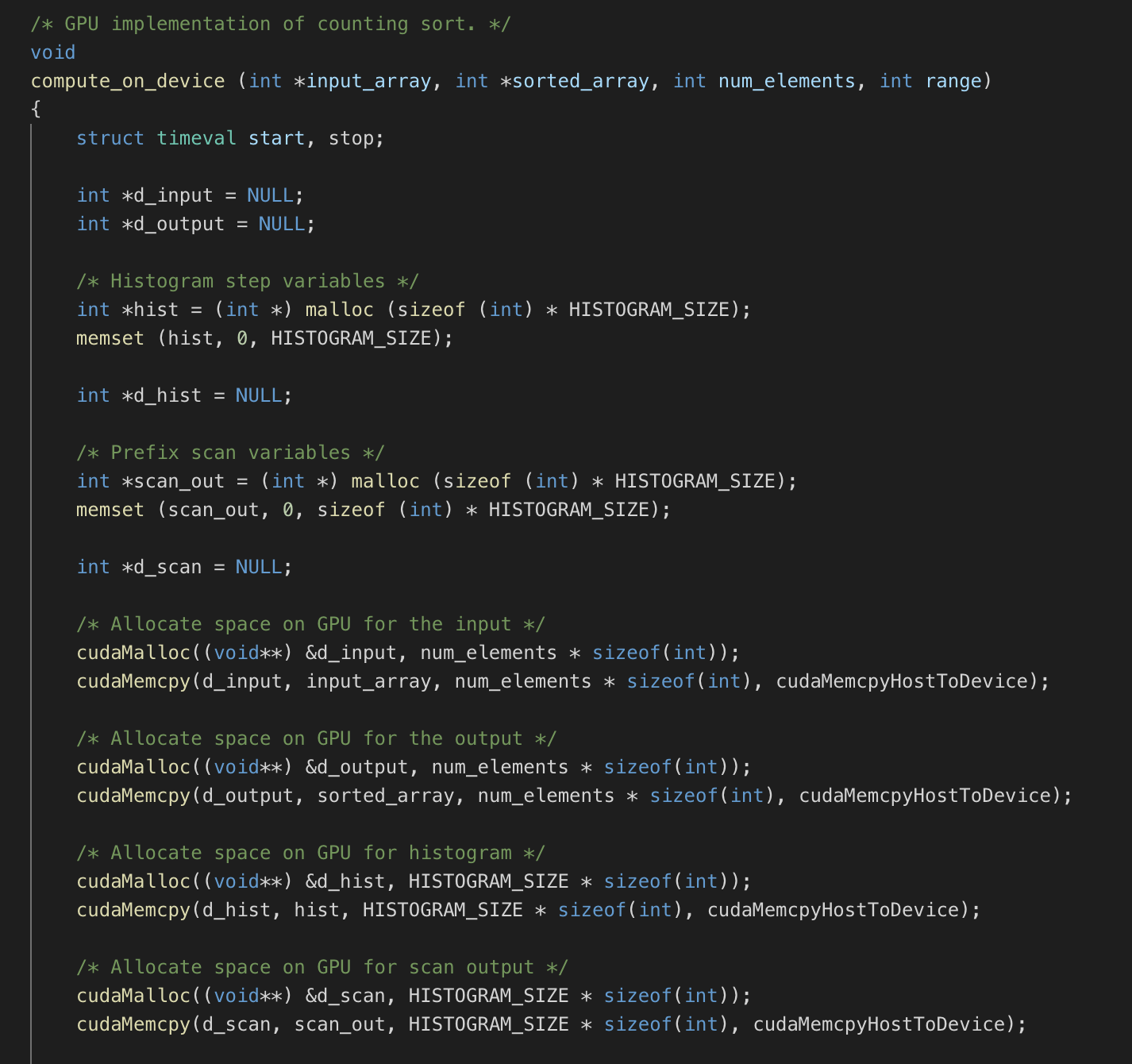
**Prof. Naga Kandasamy**

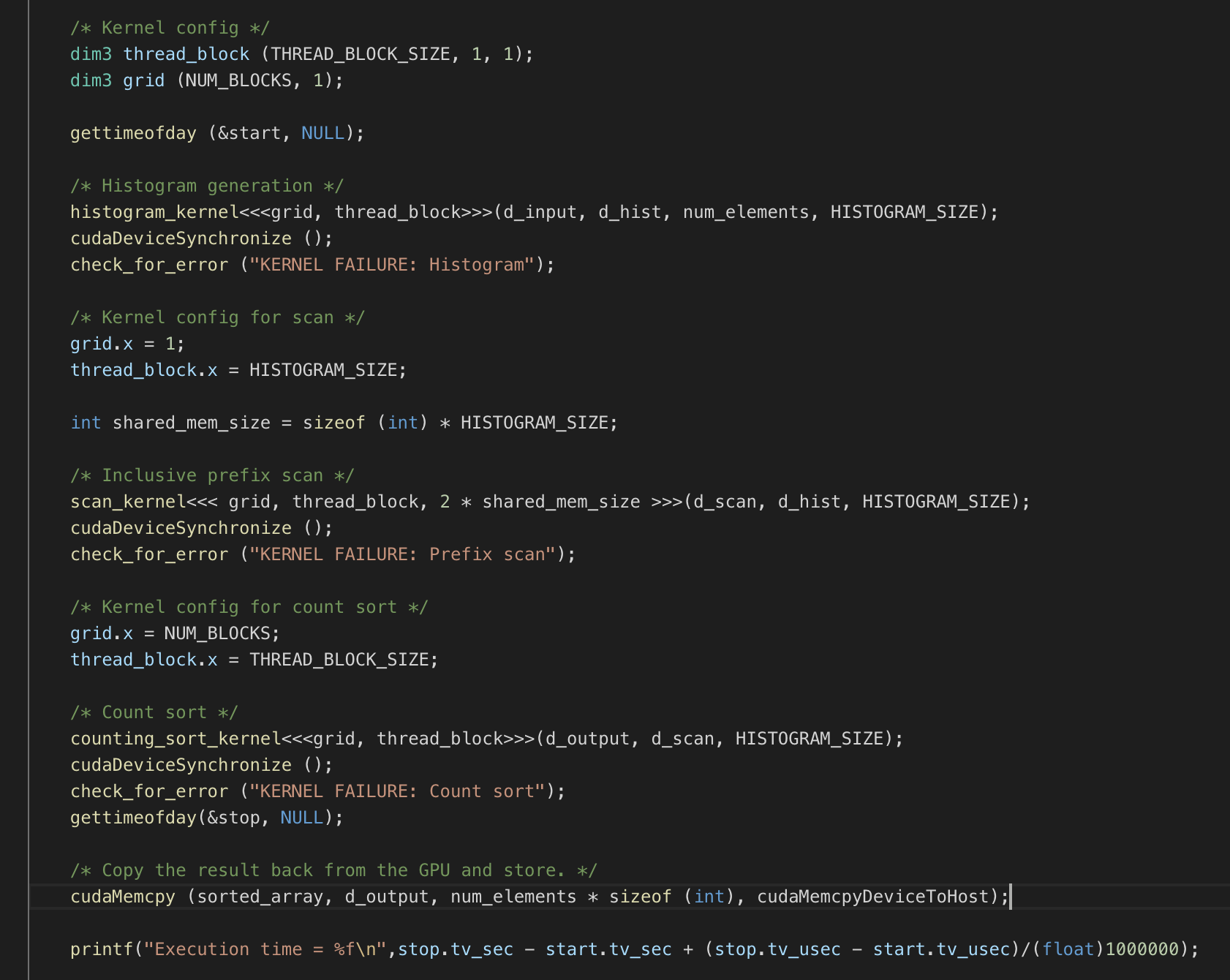
**DATE: 06/8/19**

**GPU Implementation (using CUDA)**

The first part of the CUDA implementation is to initialize the variables for the device. There are also three different kernel functions for performing the counting sort process. One for the histogram generation, another for the inclusive prefix scan, and finally one for the sorting. The kernels for the histogram generation and the final sort use the same kernel configuration, while the kernel for the prefix scan has a different one.

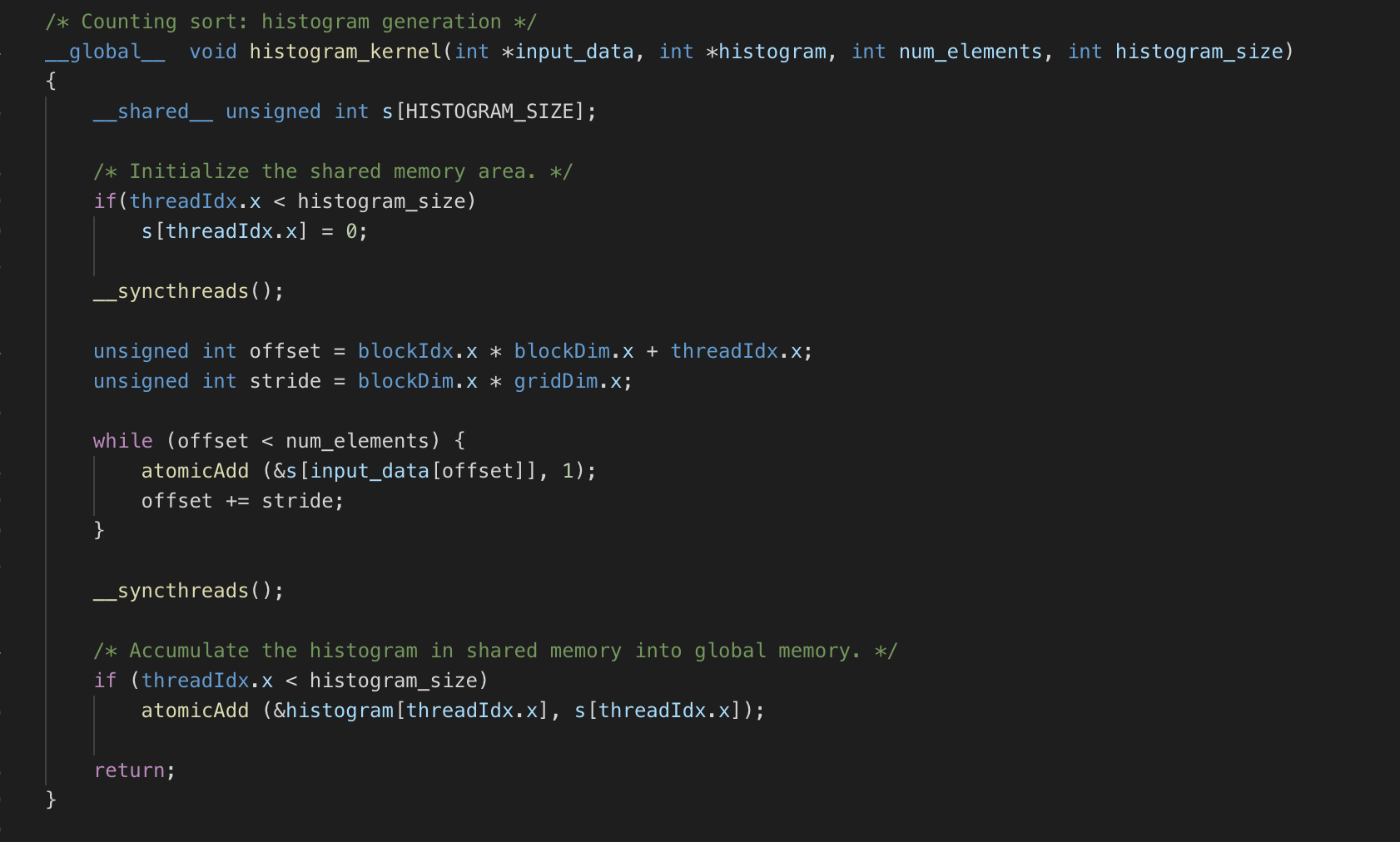
The initialization code is as follows:

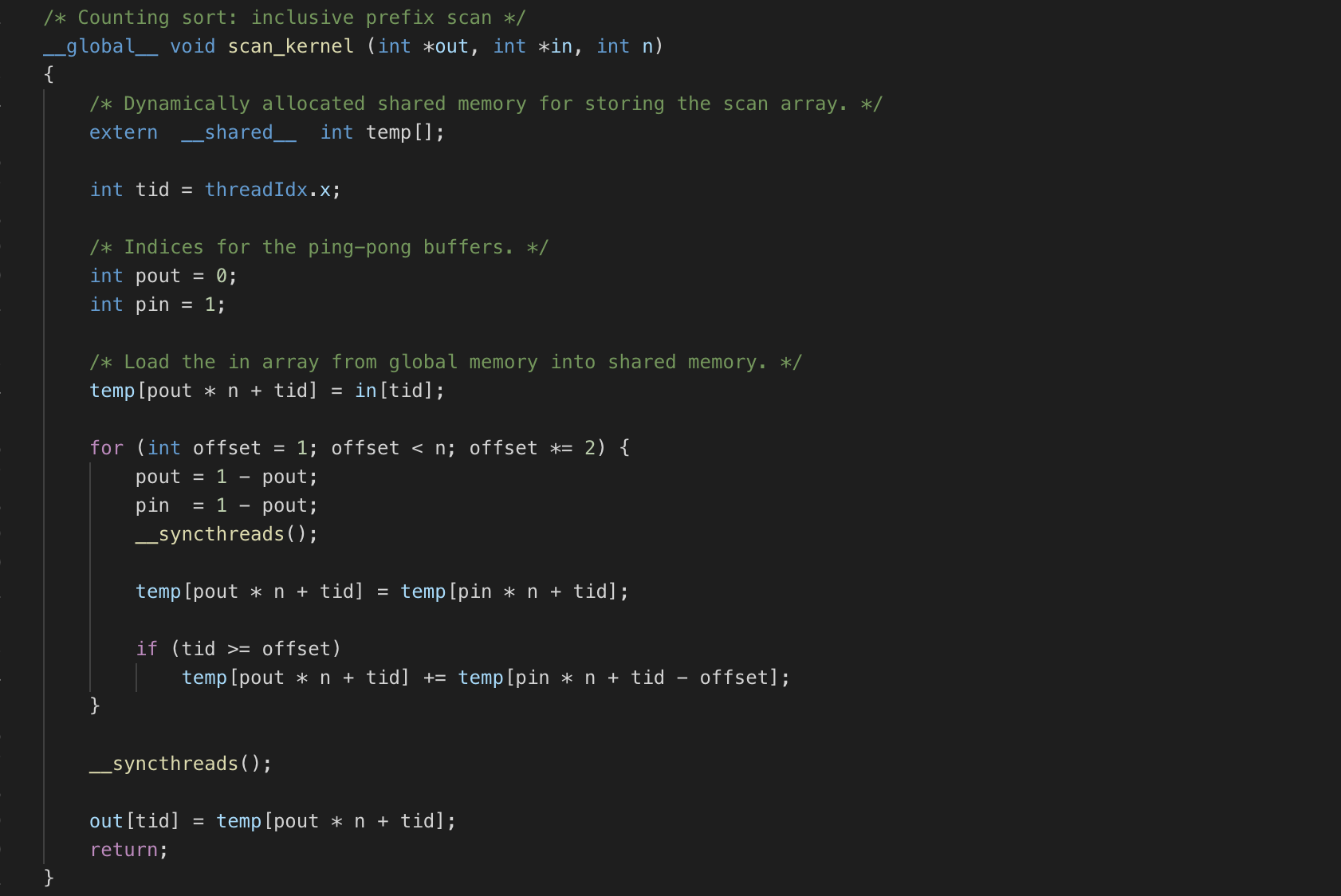
****

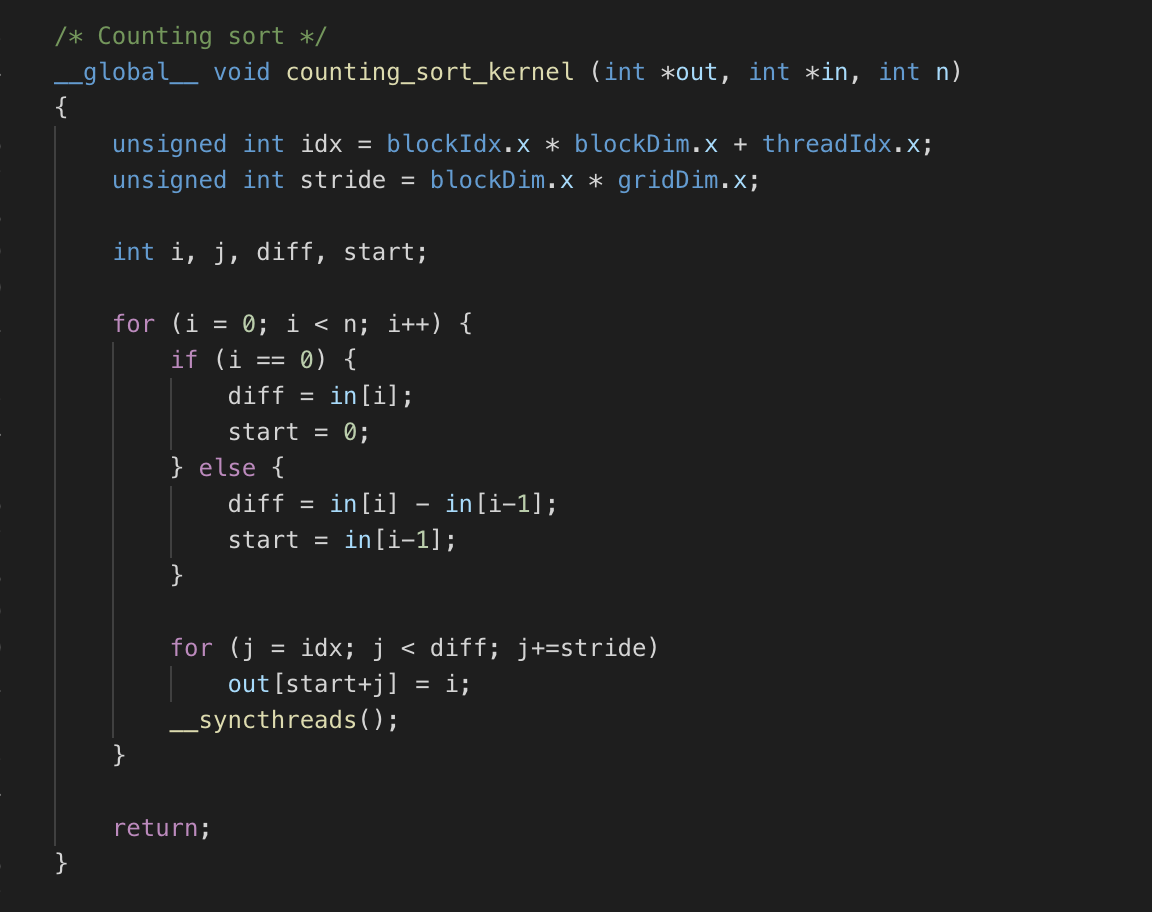
****

Regarding the kernel functions, it is possible that all three functions could’ve been combined to be just one function. However, it was more ideal to keep them all separate, which helped testing wise in order to figure out possible bugs. The histogram generation kernel utilizes the 10240 threads to quickly compute the histogram, while using shared memory and atomics. The prefix scan kernel has only 256 threads (histogram size), and also uses shared memory. Finally, the counting sort kernel has the 10240 threads, and each thread is responsible for populating the final sorted array based on the difference between the current position and the one right before, unless the current position is zero.

The kernel code is as follows:

****

****

****

**Speedup**

**Table 1: Speedup with Counting Sort on xunil-05**

|  |  |  |
| --- | --- | --- |
| **Number of Integers** | **CPU** | **GPU** |
| 1,000,000 | 3.05 ms | 0.53 ms |
| 10,000,000 | 24.66 ms | 2.14 ms |
| 100,000,000 | 201.30 ms | 20.48 ms |

*Note: xunil-05 has a Nvidia 1080 GTX GPU*

­