Lab Exercise 01 — UNASSESSED

This exercise does not carry any marks but is intended to help you get started with CUDA programming.

- First work through the *CUDA-getting-started* handout available from the Canvas page for this module.
- In `nsight`, select `File|New|CUDA C/C++ project`
- Set the project name to `VectorAdd`, untick the "`Use default location`" box and select a new directory called `VectorAdd` within your `cuda` directory, select `Executable|Empty Project` and click finish.
- Select `File|New|Source Folder` and set the folder name to `src`
- Outside `nsight`, download the `vectorAdd.cu` file from Canvas, copy it into the `src` directory you just created
- Back in `nsight`, refresh the project: press `F5`
- Create a new `Run Configuration` for this project as described in the *CUDA-getting-started* handout on Canvas.
- Click on the hammer button in the toolbar to build the project: it fails, complaining about `helper_cuda.h`

The code to use timers is inconvenient to code fully by hand. There is a set of macros that makes it significantly simpler included in the sources of the sample CUDA applications that comes with the SDK. You need to modify the project so that it finds and uses those include files. The include directory is in `samples/common/inc` under the root directory of the CUDA SDK. If you install the SDK youself using the package downloaded from NVidia for Linux, this root directory should be `/usr/local/cuda`. On the school machines, the first directory reported when you run "`module load cuda`" is the root directory.

- Select `Project|Properties|C/C++ General|Paths and Symbols`
- Under the `Includes` tab, select `CUDA C`
- Click `Add...`, tick `Add to all configurations`, Click `File system...` and select the include directory described above
- Build and run the application

Now the application is set up to experiment on.

The following is a set of questions for the Survey "Lab Exercise 01 - Unassessed" on Canvas. Please write your answers into that survey.

First enter the details requested about the machine you were working on when doing these tasks.

Task A Please enter the CPU model and the GPU model of the machine.

Task B Modify the code to calculate the speedup of the parallel version relative to 1. the host version, and 2. the single thread Device version and enter your results

Task C Try changing the value of `threadsPerBlock` on line 140. What range of values does the program still work for and what effect does it have on the timing?

Task D Try changing the `numElements` variable to significantly larger numbers: what is the largest value that the program still works on? With that largest value, what choice of `threadsPerBlock` gives the best speedup?