All exercises must be done by yourself. You may discuss questions and potential solutions with your classmates, but you may not look at their code. If in doubt, ask the instructor.

Acknowledge all sources you found useful.

Your code should compute the correct results.

Partial credit is available, so attempt all exercises.

Submit your answers as a PDF file.

The compressed archive (e.g. ZIP) file you upload to Blackboard should have your name in the filename, e.g. `JRandomStudentA4.zip`

Exercise 1

Consider the `koggestone` program that is supplied. It implements a Kogge–Stone adder to compute a sum using a tree reduction for arrays of size at most size 32.

You invoke it as:

```
$ make koggestone # only once
$ ./koggestone 5 1 8 3 2
Read 5 numbers.
Sum of 5 numbers: 19
```

1. Modify the koggestone program so that it does not use shared memory or `__syncthreads`. Use `__shfl` to communicate between threads instead. Save this as `koggestone_shfl.cu`.

2. Modify the koggestone program so that it uses `__shfl_up` to communicate between threads instead of `__shfl`. Save this as `koggestone_up.cu`.

3. Read the documentation for `__syncwarp()` at `http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html` (Section B.6). Starting from CUDA 9, where will you need to insert `__syncwarp()` in the answers to 1 and 2 to ensure correct execution?

Exercise 2

I've supplied a version of the SSSP NearFar algorithm (a simplification of the delta-stepping SSSP algorithm) to compute the single source shortest paths for a graph on the GPU. Unfortunately, this code is missing necessary synchronization in the GPU code (although the CPU code is correct) and so it computes incorrect results.

I've supplied graphs and their associated correct reference outputs as text files. Use the `test.sh` script to test for correctness. This runs `diff` on the output of `sssp_nf` and reports if it is identical (i.e. correct) or if it differs (i.e. wrong).

1. Insert the correct synchronization in code so that it computes correct results. Do not change names of any files. In your report, note your changes and your reasons for inserting the synchronizations.

2. In the worklist class, observe that `sz` is a plain `int` whereas `ndx` is a pointer to `int`. Explain why. In particular what would happen if `ndx` was a plain `int`? [Hint: Recall C/C++ argument passing semantics or come by and talk to me]

3. Which arrays in `GPUSimpleCSRGraph` (inherited from `SimpleCSRGraph`) are suitable for use as textures in the SSSP algorithm? Why? [You do not have to implement this. You may not discuss this solution with people other than the instructors.]

END.