Problem Statement

In this exercise you are supposed to do some research on the performance of different parallel sorting algorithms using programmable GPUs. The exercise has three major steps as follows.

**Step 1: Porting the CPU Parallel-Sort to CUDA**

As the first step, you need to port the parallel sort algorithm you developed in the first exercise for CPUs to the CUDA environment. That may require a complete revision of the algorithm in order to make it more efficient to run on a GPU. You may also come up with a brand-new sort algorithm on CUDA. The idea is to get as much performance as possible using conventional sorting algorithms.

**Step 2: Analyze Sorting Network Algorithms**

In this step, your task is to read, understand, and analyze a set of sorting network algorithms that are developed for the CUDA environment. You find the implementation of these algorithms under the `sortingNetworks` directory in your CUDA SDK directory. Also, you may look at a very good tutorial on the topic of sorting networks at the following URL:

http://www.iti.fh-flensburg.de/lang/algorithmen/sortieren/networks/indexen.htm
You need to understand the existing CUDA implementations of both the bitonic sort and the odd-even merge sort algorithms, and to be able to explain why the variations of these algorithms have different performance. The ultimate goal of this step is to identify the fastest variation of the sorting algorithm you have (either the conventional sorting algorithm you developed or one of the sorting network algorithms you borrowed from the CUDA SDK), and explain the reasons behind its superior performance.

**Step 3: (Optional) Propose and Implement Performance Optimizations**

In this step, having done the two previous steps, you are supposed to come up with performance optimization techniques to beat the performance of the fastest sorting algorithm you have identified in step two. The optimizations can be simple implementation tricks or major algorithmic revisions. You are free to choose any other sorting algorithm you find on the Internet as the basis for your optimization. *The basic rule for getting the bonus grade of this step is that you propose and implement some performance optimizations on your own.*

**Deliverables**

1. The source code, the `makefile`, and the compilation instructions. Your program should be compiled on Linux. Also, it’s recommended that you use the NVIDIA_CUDA_SDK project template as a basis for organizing your code.

2. A performance evaluation report containing the following items:
   a. The performance result of your own parallel algorithm
   b. The comparison between different versions of sorting network algorithms
   c. Your detailed explanation for each performance result: this explanation must be based on detailed architectural GPU features and limitations we discussed in the class (examples: GPU occupancy, global memory coalescing, shared memory access, bank conflicts, branch diversions, etc.). *So for example, just saying algorithm A is faster than algorithm B because it executes less number of instructions will not be sufficient.*
   d. An explanation of your own performance optimization tricks and techniques and your analysis of their performance impacts.