Lab#4 Parallel Pattern - Reduction

IMPORTANT:

- **Write collaborators name, if any.**
- **Test with different input sizes before submitting.**
- **Only submit ONE zip file (`FirstName_LastName_Lab4.zip`) that includes `report.pdf` and `other source files`.**

Due:

Nov. 13 11:59:59pm, 2023

If you are late (even by a minute – or heaven forbid, less than a minute late), you will receive 50% of your earned points for the designated grade as long as the assignment is submitted by 11:59pm the following day, based on the due date listed on the above and confirmed by the instructor. If you are more than 24 hours late, you will receive a zero for the assignment and your assignment will not be graded at all.

Goal:

The objective of this Lab is to get you familiar with the parallel reduction algorithm.

Instructions

1. Download and extract lab4-reduction.zip from eLC. The folder should contain 5 files: **Makefile, reduction_kernel.cu, reduction_main.cu, support.cu, support.h**. Carefully study the code and ask questions on eLC if you have any. Run the `make` command to compile your files.
2. If you have any questions about connecting to a remote machine, please refer to the Lab0-setup manual.

3. Look for the statement “INSERT YOUR CODE HERE”. Test different thread block sizes and `choose the one that works best for your case`. The performance will be an important factor for your final grade.
4. Your program should be able to accept valid input size as arguments. Check the code carefully before starting.

Testing

After you are done with coding, answer the following questions and submit the `report (PDF)` and `5 source files` (all in one zip file) in the eLC:

1. Which CUDA machines have you tested?
2. Can your program compile properly?
3. Is your program working correctly and did it pass the test?
4. How many times does a single thread block synchronize to reduce its portion of the array to a single value?
5. What is the minimum, maximum, and average number of "real" operations that a thread will perform? "Real" operations are those that directly contribute to the final reduction value.

Grading

1. Your submission will be graded based on the report and code (including but not limited to code quality, correctness, performance, and readability).
2. Others
   - **Functionality/knowledge: 65%**
     - Correct code and output results
     - Correct usage shared and constant memory to cover global memory access latency
     - Correct handling of boundary cases
   - **Answers to question: 35%**
     - Correct answer to questions

- Sufficient work shown
- Neatness, clarity, and efficiency