CS433: Assignment#3 (Due by 14th November 2024 midnight)

1. In this problem, you will write a CUDA program for the Gauss-Seidel iterative grid solver. You will use the problem description as discussed in the class. The program should accept two command line arguments, namely the grid dimension n for an $n \times n$ grid and the number of threads t. Your program will be tested for only powers of two n and t. The program should output the number of iterations needed to converge, the average difference per grid element at the end, and the time taken to converge. Your program should have a separate CUDA kernel for initializing the grid. You should not include the initialization time in your time measurement. You should report results for a random initialization as was done in the class demo for OpenMP/MPI. Use a tolerance of 1e-5 for convergence and an iteration count limit of 1000 as used in the class demo. First, develop a basic working CUDA program. For any grid-wide synchronization, you are free to choose the method of synchronization. We have discussed three options in the class: cooperative thread groups, sense-reversing barrier, and multiple kernel launches. Ensure that your program does not deadlock for any number of threads. Second, optimize the basic program using tree reduction. Third, optimize it further using shared memory. For evaluation, first choose the largest n that you can run within reasonable time bound. Next choose the best thread hierarchy configuration for your GPU and the basic program for this n. Prepare a table showing how performance of the basic program varies with thread count t. Next, for the best performing thread count, show the effect of the two optimizations. Please feel free to apply additional optimizations; evaluate them adequately. Compare your best performing CUDA program for the grid solver with the best performing OpenMP or POSIX thread program (use the best thread count for your CPU in this comparison). You can directly use the demo OpenMP/POSIX thread program developed in the class for this purpose. **(60 points)**

2. In this problem, you will write a CUDA program to compute single-precision floating point (datatype `float`) matrix-vector product. Given an $n \times n$ matrix A and an $n \times 1$ vector x, the matrix-vector product computes $y = Ax$. The program should accept two command line arguments, namely the matrix dimension n and the number of threads t. Your program will be tested for only powers of two n and t. Your program should have a separate CUDA kernel for initializing A and x. You should not include the initialization time in your time measurement. The host program at the end should compute the average absolute difference between y_i^{CPU} and y_i^{GPU} for $0 \leq i < n$ and print it along with the measured time. Exclude the average difference computation time from the measured time. First, develop a basic working CUDA program. Second, optimize the basic program using shared memory. For evaluation, first choose the largest n that you can run within reasonable time bound. Next choose the best thread hierarchy configuration for your GPU and the basic program for this n. Prepare a table showing how performance of the basic program varies with thread count t. Next, for the best performing thread count, show the effect of the shared memory optimization. Please feel free to apply additional optimizations; evaluate them adequately. Compare your best performing CUDA program with the best performing OpenMP or POSIX thread program. **(40 points)**

What to submit and how. For each question, submit the parallel program. Prepare a report describing your parallel algorithms, the optimizations you applied for improving performance, and the performance results. Explain the trends seen in the results. You will not get any marks if your program does not compile or crashes during execution. Only the programs that compile and run to completion without any error will be considered for grading. Please send your submission (two programs and a PDF report) via email to cs433autumn2024@gmail.com with subject “Assignment#3 submission Roll#X”. Replace X by your roll number in the subject line.