# EECS 587 Homework: Due at the start of class 8 November 2022

For this project you are allowed to work in teams of 2, if you wish. If you do so, make sure both names are on the project when you turn it in.

Using the GPU nodes on Great Lakes, run an efficient CUDA program to do $t$ repeated stencil calculation on an $n \times n$ matrix $A(0:n-1, 0:n-1)$ where the entries are double precision reals (i.e., 16 bytes). Do this for $n = 500$, 1000 and 2000, and $t = 10$. The same program must be used for all of the runs, and should work no matter what the initial entries of $A$ are, what $t > 1$ is, and no matter what $n > 1$ is, as long as $A$ takes less than 1/2 the RAM on the GPU.

When you submit your program you must use a time limit $\leq$ 5 minutes. You must use the Great Lakes GPUs for your timing runs but you can debug on any machine (including your own).

**Procedure**: For each $n$:

1. On the CPU initialize $A$ using the definition below, and copy it into the GPU.
2. Initialize the CUDA timer.
3. On the GPU do $t$ iterations of updating $A$, defined below.
4. Compute the 2 verification values on the GPU (see below).
5. Stop the CUDA timer.
6. Copy the verification values to the CPU's RAM.
7. Print out the elapsed time and the verification values.

Turn in the program code, the timing and verification values, and the report.

**To initialize $A$:** for all $0 \leq i, j \leq n-1$, $A(i,j) = (1 + \cos(2i) + \sin(j))^2$

**Iterative step:** In each iteration, let $A_o$ denote the previous value of $A$. Then for all $0 \leq i, j \leq n-1$, the new value of $A(i,j)$ is:

- $A_o(i,j)$ if $((i = 0) \vee (i = n-1) \vee (j = 0) \vee (j = n-1))$, i.e., it is unchanged along the border
- otherwise it is $A_o(i,j)$ plus the 2nd smallest value of $\{A_o(i+1, j+1),\ A_o(i+1, j-1),\ A_o(i-1, j+1),\ A_o(i-1, j-1)\}$. If there is a tie for smallest, then that value is also the 2nd smallest, i.e., this is viewed as a multiset, not strictly as a set.

Be careful to use $A_o$, not values from the current iteration.

**To verify a run:** The 2 verification values are based on the final value of $A$. The GPU should determine the sum of all of the entries of $A$ and the value $A(37, 47)$.

The code to compute the verification sum must be written by you and must be run on the GPU. You cannot try to find versions written by someone else, nor use predefined routines supplied by packages such as Thrust. However, if you cannot get your own code to run correctly you are allowed to use Thrust or any other suitable routine you can find, but in your report you must say that you have done this, and you will automatically lose 10% of the grade.

**The report:** Write a short report which includes a brief description of how you did the calculations, the verification values, and your timing results. Analyze how the program scales with the input size, and explain its behavior. Note that the number of cores isn't changing, so you can't analyze scaling in terms of the number of cores. The report needs to be typewritten, though you can do drawings by hand. You will be graded on correctness, the quality of your report, and achieved performance.

Depending on how long the programs are taking and the turnaround time on Great Lakes, we might adjust the matrix sizes or number of iterations.