CS6023: GPU Programming
Assignment 1

Problem specification

Write three separate CUDA C++ kernels for adding up two integer matrices i.e. add elements at the same *i, j* position in both the matrices. In the first kernel ***per_row_kernel***, each thread should process a complete row of the input matrices. In the second kernel ***per_column_kerne***l, each thread should process a complete column of the input matrices. In the third kernel ***per_element_kernel***, each thread should process exactly one element from both the input matrices. For the evaluation purpose, ***per_row_kernel*** will be invoked with ***1D grid and 1D blocks***, ***per_column_kernel*** will be invoked with ***1D grid and 2D blocks*** and ***per_element_kernel*** will be invoked with ***2D grid and 2D blocks***.

Input: Size of matrix (*m* and *n*) and two integer matrices **A** and **B** of same size.
Output: A matrix **C** of the same size (say *m* x *n*) as **A** and **B**, storing the result of **A+B**. **C** will be provided in the kernel function as a parameter, you need to modify that only.

Points to be noted:

- The file **kernels.h** provided by us contains the prototypes of the three kernels.
- Do NOT change the names and the signatures of the kernels provided.
- Sample input and sample output matrices are shown below. Pay attention to the position of each element in the input and the output matrices.
- The size *m* and *n*, of the input matrices used for evaluation will be in the range: $5 \leq m \leq 2^{13}$ and $5 \leq n \leq 2^{13}$.
- The updates should be performed on the **C** matrix and should finally store **A+B** as result. Do not use any intermediate matrices.
- Do not write any print statements inside the kernel.
- You can use your own main.cu to test your code.
- Test your code on large matrices.

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 11 & 12 & 13 \\ 13 & 14 & 15 & 16 \end{bmatrix} + \begin{bmatrix} 10 & 20 & 30 & 40 \\ -50 & -60 & -70 & -80 \\ 90 & 110 & 120 & 130 \\ 130 & 140 & 150 & 160 \end{bmatrix} = \begin{bmatrix} 11 & 22 & 33 & 44 \\ -45 & -54 & -63 & -72 \\ 99 & 121 & 132 & 143 \\ 143 & 154 & 165 & 176 \end{bmatrix}$$

A **B** **C**

Figure 1: Sample input and output matrices

Learning suggestions:

- Write a CPU-version of code achieving the same functionality. Time the CPU code and GPU code separately for large matrices and compare the performances