CSCI-GA.3033-025
Graphics Processing Units (GPUs): Architecture and Programming
Programming Assignment 1

In this first programming assignment, we will implement a simple integer matrix-vector multiplication. We will multiply a square matrix of dimension mxm with a vector of dimension m. To have a point of comparison, you will implement a sequential version and a GPU version.

To measure the time taken by a piece of code, use the following setup:

```
#include <time.h>
…
double time_taken = 0;
clock_t start, end;

start = clock();
(the code you want to measure)
end = clock();  // end of measuring
time_taken = ((double)(end-start)) / CLOCKS_PER_SEC;
```

For sequential version, the code is the operations done for the matrix-vector multiplication.

For CUDA version, the code to be measured is:

- Sending data to the device
- Kernel to do the computations.
- Bringing data from the device

Design your program as follows:

- Filename: **matrixvector.cu**
- Compile with: nvcc -o matrixv matrixvector.cu
- To execute, the user needs to type: ./matrixv *num blocks threads* where *num* is the dimension of the square matrix and the vector, *blocks* is the total number of blocks for the CUDA version, and *threads* is the number of threads per block.
- Your program must first execute the sequential version and print its execution time as: **sequential version: …. seconds**
- Then, your program executes the GPU version and prints: **GPU version: …. seconds**
- A look at the code vecadd.cu given on the course website next to lecture 3 will be helpful.

Experiments:

You will test your program with the following dimensions: 256, 512, 1024, 4096, 8192, 16384, 32768, 65536, 131072, and 262144.

For each dimension, do the following experiments:

- 4 blocks of 64 threads each
- 8 blocks of 32 threads each

Put the results in a table where the rows represent the dimensions, and the columns are the two configurations. The table cells contain the speedup: **(sequential time) / (CUDA time).**

Based on the table you formed above, answer the following questions:

1. What pattern do you see when the problem size increases?
2. Please explain why we see this pattern.
3. Which configuration has a better performance on average: more blocks of smaller sizes, or less blocks of bigger sizes (i.e. larger number of threads)?
4. Justify your answer to the above question.

Submission:

- Put the table and your answers in a pdf file called: **netID.pdf** where netID is your own netID.
- Put matrixvector.cu and netID.pdf in one zip file with name: **lasname.firstname.zip** where these are your last name and first name.
- Submit the zip file through Brightspace before the deadline.
- You can submit several times till the deadline. We will grade only the last version.