CMPS 297S/396AA: GPU COMPUTING
ASSIGNMENT 4

In this assignment, you will implement a convolution kernel that uses shared memory tiling. Your kernel is expected to work for any set of input dimensions so make sure to handle boundary conditions correctly. You must use shared memory tiling to receive credit for this assignment.

Instructions

1. Place the files provided with this assignment in a single directory. The files are:
   - `main.cu`: contains setup code, sequential code, and naïve kernel code without tiling
   - `kernel.cu`: where you will implement your code (you should only modify this file)
   - `common.h`: for shared declarations across `main.cu` and `kernel.cu`
   - `timer.h`: to assist with timing
   - `Makefile`: used for compilation

2. Edit `kernel.cu` where TODO is indicated to implement the following:
   - Copy the convolution mask to constant memory
   - Configure and call the kernel
     - Be careful: you need enough threads per block to load an entire *input* tile, but enough blocks in the grid to process every *output* tile
   - Perform the computation in the kernel:
     - Find the *output* element that each thread is computing
       - Be careful: blockDim corresponds to the *input* tile size
     - Find the *input* element that each thread in the block is loading
     - Use all the threads in the block to load the *input* tile to shared memory
     - Use a subset of the threads in the block to compute the *output* tile

3. Compile your code by running: `make`

4. Test your code by running: `./convolution`
   - If you are using the HPC cluster, do not forget to use the submission system. Do not run on the head node!
   - For testing on different input sizes, you can provide your own values for the input dimensions as follows: `./convolution <height> <width>`

Submission

Submit your modified `kernel.cu` file via Moodle by the due date. Do not submit any other files or compressed folders.