EECS 471 Assignment 4
3D Convolution

Due Date: Friday, February 17 at 11:59 pm

Remember to submit on Gradescope as well!

Objective

The purpose of this lab is to implement a 3D convolution using constant memory for the kernel and 3D shared memory tiling.

Instructions

- Edit the code to implement a 3D convolution with a 3x3x3 kernel in constant memory and a 3D shared-memory tiling.
- Edit the code to launch the kernel you implemented. The function should launch 3D CUDA grid and blocks. You may use any of the 3 tiling strategies taught in class.

Algorithm Specification

You will be implementing the following 3D convolution.

```
for z_out = 0 to z_size - 1: {
  for y_out = 0 to y_size - 1: {
    for x_out = 0 to x_size - 1: {
      let res = 0;
      for z_mask = - MASK_RADIUS to MASK_RADIUS: {
        for y_mask = - MASK_RADIUS to MASK_RADIUS: {
          for x_mask = - MASK_RADIUS to MASK_RADIUS: {
            let z_in = z_out + z_mask;
```

```
            let y_in = y_out + y_mask;
            let x_in = x_out + x_mask;

            // Pad boundary with 0

            if (z_in >= 0 && z_in < z_size &&

                  y_in >= 0 && y_in < y_size &&

                  x_in >= 0 && x_in < x_size) then {

              res += mask[z_mask + MASK_RADIUS][y_mask +
                  MASK_RADIUS][x_mask + MASK_RADIUS] *
                  in[z_in][y_in][x_in]

            }

          }

        }

      out[z_out][y_out][x_out] = res;

    }

  }

}
```

- The kernel size is fixed to 3x3x3, given `MASK_WIDTH = 3` and `MASK_RADIUS = 1`.
- Halo elements should be read as 0.
- You should support input data of any size.
- Note that the input and output size is the same.

Other Notes

The raw format of the input data is a flattened array, where the *first three* elements appended onto the front of the flattened array are the `z_size`, `y_size`, and `x_size` respectively. For example, a 5x4x3 input array will look like

```
float inputData[] = { 5.0, 4.0, 3.0, ... < 60 floats > }
```

Without the first three elements considered, a point (z,y,x) in a flattened 3D array may be accessed at index `z * (y_size * x_size) + y * (x_size) + x`.
The template code reads the first three elements into `z_size`, `y_size`, and `x_size`. You will need to copy the rest of the data to the device.

Likewise, the result needs to have the sizes prepended to check your result correctly. The template code does that as well, but you must copy the data into `outputData` from the fourth element on.

Remember that you can get a pointer to the fourth element of the array with `&array[3]`.

Plagiarism

Plagiarism will not be tolerated. The first offense will result in the two parties getting a 0 for the programming assignment. Second offense results in a 0 for the course.