**Lab 5 Lab Report: Shubh Patel**

1. Describe all optimizations you tried regardless of whether you committed to them or abandoned them and whether they improved or hurt performance.

* The biggest optimization I did was using shared memory. For each thread block, I had a privatized bin which improved performance. But with that in mind, this introduced another parameter, the block size. To try to measure which block size is most adequate, I believe I would need a much bigger dataset as the differences between the times of each block size are arbitrary. And having more blocks means more shared memory allocation.
* Another optimization I tried to do was having the same number of threads in the grid as bins instead of array elements. But the timings were too close to decide which one was better, so I reverted back to array elements and that having the number of threads in a grid equal the number of array elements is more generalized.

2. Were there any difficulties you had with completing the optimization correctly?

* Yes, I had some trouble converting the logic from the lecture slides to adapt to the way I had mine. If you look at the lecture slides, in the kernel, there was an if statement for zeroing out the private bins, a loop to add to the private bin, and an if statement for add back to global memory. But for mine, I had to change it to have a loop for zeroing out bins, an if statement to add to the private bin, and a loop to add it back to global memory.

3. Which optimizations gave the most benefit?

* I believe utilizing shared memory is most beneficial as privatization doesn’t lock up all the threads in a grid just waiting for 1 thread to finish its atomic operation.

4. For the histogram kernel, how many global memory reads are being performed by your kernel? Explain.

* In my kernel, the number of global memory reads equals the number of elements. I have 1 thread per element and each thread performs 1 global memory read.

5. For the histogram kernel, how many global memory writes are being performed by your kernel? Explain.

* In my kernel, the number of global memory writes equals (Bin\_Size / BlockDim.x) \* (Total number of threads in the grid). So basically, in my kernel, I have 1 global memory write. But that write is in a for loop. And since all threads in a block collaborate together, each thread executes a global memory write (Bin\_Size / BlockDim.x) times. I get this number because in the for loop, the each index increases by BlockDim.x each iteration until it hits Bin\_Size.

6. For the histogram kernel, how many atomic operations are being performed by your kernel? Explain.

* The number of atomic operations being performed by my kernel = (# of threads in the grid) + ((Bin\_Size / BlockDim.x) \* (Total number of threads in the grid)). Or for each thread = 1 + (Bin\_Size / BlockDim.x). I got these numbers because 1 atomic operation is used for a global memory read, and the other is used for the global memory write. So I just combined those two together.

7. For the histogram kernel, what contentions would you expect if every element in the array has the same value?

* I’d expect heavy contentions for both global memory and shared memory. Because all the threads would be trying to increment the same bin.

8. For the histogram kernel, what contentions would you expect if every element in the input array has a random value?

* I’d expect light contentions for both global memory and shared memory. While there is the possibility of some elements being the same, the contention would still be significantly less than if all the elements for the same.