ASSIGNMENT - 4

Name - Srushti Hembade
Student Id - 862395839
Net id - shemb001

Assume you perform a histogram operation on an array of size N, 16 blocks, block size of B and 1024 bins. Assume a privatized histogram implementation.

Question 1: How many global memory reads are performed during the histogram kernel?
→ N + 1024 * 16
When threads contact the input array to get elements for the histogram computation, N stands for these global memory reads.
When threads accumulate the private histograms into the global histogram bins, 1024x16 depict for these global memory reads. During the accumulation stage, each of the 16 blocks reads data from 1024 bins in global memory.
So we add these two ,reading and writing to the global histogram bins while accumulation and retrieving input components for the histogram computation of the total global memory reads.

Question 2: How many global memory writes are performed during the histogram kernel?
→ 1024 * 16
When threads update the global histogram bins, 1024 x 16 represents the writes to global memory. During the accumulation stage, each of the 16 blocks writes to 1024 bins in global memory.

Question 3: How many total atomic operations are performed?
→ N + 1024 * 16
The atomic actions that occur when threads update the shared memory's private histograms are represented by N. For a given bin in its private histogram, each thread increments the count using an atomic transaction.
The atomic actions that occur when threads aggregate private histograms into the global histogram bins are explained by 1024×16. During the accumulation stage, each of the 16 blocks completes 16 atomic operations (1024 reads and 1024 writes).

OUTPUT screenshot for the Histogram (no arguments) -

```
bender /home/cegrad/shembade/histogram-srushtih1 $ ls
kernel.cu  main.cu  Makefile  README.md  support.cu  support.h
bender /home/cegrad/shembade/histogram-srushtih1 $ make
nvcc -c -o main.o main.cu -O3
nvcc -c -o support.o support.cu -O3
nvcc main.o support.o -o histogram -lcudart
bender /home/cegrad/shembade/histogram-srushtih1 $ ls
histogram  kernel.cu  main.cu  main.o  Makefile  README.md  support.cu  support.h  support.o
bender /home/cegrad/shembade/histogram-srushtih1 $ ./histogram &> outfile
bender /home/cegrad/shembade/histogram-srushtih1 $ cat outfile

Setting up the problem...0.018901 s
    Input size = 1000000
    Number of bins = 4096
Allocating device variables...0.097009 s
Copying data from host to device...0.000992 s
Launching kernel...0.079371 s
Copying data from device to host...0.000033 s
Verifying results...0: 256/256, 1: 257/257, 2: 239/239, 3: 229/229, 4: 238/238, 5: 236/236, 6: 23
3/253, 11: 211/211, 12: 244/244, 13: 264/264, 14: 253/253, 15: 249/249, 16: 230/230, 17: 241/241,
```

```
/226, 4054: 249/249, 4055: 236/236, 4056: 240/240, 4057: 259/259, 4058: 252/252, 4
 235/235, 4064: 262/262, 4065: 202/202, 4066: 269/269, 4067: 246/246, 4068: 245/24
073: 238/238, 4074: 253/253, 4075: 271/271, 4076: 259/259, 4077: 223/223, 4078: 25
1, 4083: 233/233, 4084: 253/253, 4085: 227/227, 4086: 216/216, 4087: 260/260, 4088
7/257, 4093: 220/220, 4094: 254/254, 4095: 214/214,
TEST PASSED
```

OUTPUT screenshot for the Histogram (one arguments - 2048) -

```
bender /home/cegrad/shembade/histogram-srushtih1 $ ./histogram 2084 &> outfile1
bender /home/cegrad/shembade/histogram-srushtih1 $ cat outfile1

Setting up the problem...0.000098 s
    Input size = 2084
    Number of bins = 4096
Allocating device variables...0.103849 s
Copying data from host to device...0.000064 s
Launching kernel...0.003428 s
Copying data from device to host...0.000028 s
Verifying results...0: 1/1, 1: 0/0, 2: 1/1, 3: 0/0, 4: 0/0, 5: 1/1, 6: 0/0, 7: 0/0, 8: 0/
5: 0/0, 16: 0/0, 17: 1/1, 18: 0/0, 19: 2/2, 20: 0/0, 21: 0/0, 22: 1/1, 23: 1/1, 24: 1/1,
 1/1, 32: 0/0, 33: 2/2, 34: 0/0, 35: 0/0, 36: 0/0, 37: 2/2, 38: 0/0, 39: 0/0, 40: 0/0, 41
/0, 48: 0/0, 49: 1/1, 50: 0/0, 51: 1/1, 52: 0/0, 53: 1/1, 54: 1/1, 55: 1/1, 56: 0/0, 57:
, 64: 1/1, 65: 0/0, 66: 1/1, 67: 0/0, 68: 4/4, 69: 0/0, 70: 0/0, 71: 0/0, 72: 0/0, 73: 2/
80: 0/0, 81: 3/3, 82: 0/0, 83: 1/1, 84: 1/1, 85: 0/0, 86: 0/0, 87: 0/0, 88: 0/0, 89: 0/0,
```

```
041: 1/1, 4042: 3/3, 4043: 0/0, 4044: 1/1, 4045: 1/1, 4046: 0/0, 4047: 1/1, 4
: 1/1, 4055: 0/0, 4056: 2/2, 4057: 0/0, 4058: 2/2, 4059: 1/1, 4060: 0/0, 4061
/1, 4068: 1/1, 4069: 1/1, 4070: 2/2, 4071: 0/0, 4072: 1/1, 4073: 0/0, 4074: 0
 4081: 0/0, 4082: 0/0, 4083: 2/2, 4084: 1/1, 4085: 0/0, 4086: 0/0, 4087: 1/1,
94: 0/0, 4095: 0/0,
TEST PASSED
```

OUTPUT screenshot for the Histogram (2 arguments - 2048, 1024) -

```
bender /home/cegrad/shembade/histogram-srushtih1 $ ./histogram 2084 1024 &> outfile2
bender /home/cegrad/shembade/histogram-srushtih1 $ cat outfile2

Setting up the problem...0.000097 s
    Input size = 2084
    Number of bins = 1024
Allocating device variables...0.101950 s
Copying data from host to device...0.000105 s
Launching kernel...0.003983 s
Copying data from device to host...0.000017 s
Verifying results...0: 4/4, 1: 4/4, 2: 4/4, 3: 4/4, 4: 0/0, 5: 2/2, 6: 3/3, 7: 1/1, 8: 1/1, 9: 0/
5: 0/0, 16: 0/0, 17: 3/3, 18: 3/3, 19: 3/3, 20: 2/2, 21: 1/1, 22: 4/4, 23: 1/1, 24: 3/3, 25: 2/2,
 2/2, 32: 0/0, 33: 4/4, 34: 4/4, 35: 2/2, 36: 1/1, 37: 2/2, 38: 0/0, 39: 2/2, 40: 1/1, 41: 1/1, 4
/1, 48: 1/1, 49: 2/2, 50: 2/2, 51: 2/2, 52: 1/1, 53: 3/3, 54: 1/1, 55: 4/4, 56: 1/1, 57: 1/1, 58:
```

```
958: 1/1, 959: 1/1, 960: 3/3, 961: 2/2, 962: 1/1, 963: 0/0, 964: 3/3, 965: 2/
/1, 973: 5/5, 974: 1/1, 975: 2/2, 976: 3/3, 977: 5/5, 978: 3/3, 979: 3/3, 980
7: 4/4, 988: 0/0, 989: 1/1, 990: 1/1, 991: 2/2, 992: 2/2, 993: 3/3, 994: 5/5,
/3, 1002: 0/0, 1003: 2/2, 1004: 2/2, 1005: 2/2, 1006: 2/2, 1007: 4/4, 1008: 1
 1015: 4/4, 1016: 2/2, 1017: 2/2, 1018: 2/2, 1019: 2/2, 1020: 4/4, 1021: 2/2,
TEST PASSED
```