ECE 408 / CS 483 / CSE 408
Applied Parallel Programming
Second Exam, Spring 2020

ANY 3-HOUR BLOCK OF YOUR CHOICE
between release (6 p.m. CDT on Monday 27 April 2020)
and 6 p.m. CDT on Wednesday 29 April 2020

- **Use only the associated text file to provide your solution.**
- **Use no more than one line for each answer (you should not need more).**
- **Do not modify the text file in any way other than to add your answers.**
- **Send your completed solution to lumetta@illinois.edu**
    - **from your campus email (netid@illinois.edu)**
    - **before the deadline (6 p.m. CDT on Wednesday 29 April 2020)**
- **This is an open resource exam. You may use your book, a calculator, the class web page, or any other static resource on the Internet.**
- **You may NOT in any way communicate about the content of the exam with any human other than Prof. Lumetta before turning in your solution.**
- **Please avoid communicating with any other student about the content of the exam until after the deadline has passed, even if both of you have already submitted your solutions.**
- **Don't panic, and good luck!**

Problem 1	14 points	______________________________
Problem 2	27 points	______________________________
Problem 3	26 points	______________________________
Problem 4	24 points	______________________________
Problem 5	9 points	______________________________
Total	100 points	______________________________

Problem 1 (14 points): Short Answer Questions

PLEASE SELECT EXACTLY ONE ANSWER FOR EACH QUESTION.

1. **(3 points)** Which of the following statements is **TRUE** of the PCI/PCIe interface over which CPUs typically communicate with GPUs?

   A. PCIe offers higher bandwidth than most CPU memory interfaces, making GPUs an attractive way to improve both computation and memory speeds.

   B. PCIe uses parallel signaling to increase peak connection bandwidth.

   C. The need to support antiquated PCI devices implies that PCIe will never be able to provide adequate bandwidth for host-device communication.

   D. Each PCIe connection is arbitrated by controller hardware that determines which end of the connection is allowed to provide data at any point in time.

   E. The 8b/10b encoding used by PCIe makes the physical layer somewhat inefficient, as it adds 25% overhead to transmitted data.

2. **(3 points)** Which of the following statements is **TRUE** of modern CPU-GPU architectures?

   A. Use of pinned memory accelerates memory copy operations from host memory to GPU memory, but does not benefit memory copy operations from GPU memory to host memory.

   B. Use of pinned memory for host-side data structures accelerates copying of data between host memory and device memory by registering the pinned locations with the GPU.

   C. Use of pinned host memory enables memory to be accessed in larger blocks, thus improving the throughput of memory copy operations.

   D. Use of pinned memory for host-side data structures accelerates copying of data between host memory and device memory by reducing the number of copy operations needed.

   E. All host-side data structures should be allocated in pinned memory to minimize the time spent copying data to the GPU.

3. **(3 points)** Which of the following statements is **TRUE** of the alternative programming models that we discussed in lecture?

   A. OpenACC targets code development for systems with hundreds of GPUs operating in parallel by abstracting the notion of thread blocks into gangs and workers.

   B. With MPI, each GPU becomes one node within a pool of computing resources called MPI_COMM_WORLD. Each node then executes the same program in an SPMD fashion.

   C. Information about parallelism in an OpenACC program is communicated solely through #pragma statements, guaranteeing that compilation for a sequential CPU produces correct results.

   D. Each process in an MPI run receives its own unique identifier, much like the thread indices in a CUDA program.

   E. Enabling network interface cards to address GPU memory directly might accelerate MPI programs making use of GPUs by allowing one process to write into another process' GPU memory before the receiving process has called `MPI_Recv` to receive the message.

Problem 1, continued:

PLEASE SELECT EXACTLY ONE ANSWER FOR EACH QUESTION.

4. **(3 points)** Consider the following two statements:

   I. For a sparse matrix-vector multiply computation using the JDS-Transposed format (and single-precision float input values), values in the output vector are likely to differ between a simple CPU implementation and a correct parallel implementation.

   II. For an inclusive prefix sum (scan) on an array of single-precision floating-point input values, values in the output vector are likely to differ between a simple CPU implementation and a correct parallel implementation.

   A. Statement I is true; but Statement II is false.

   B. Statement I is false; but Statement II is true.

   C. Both statements are true.

   D. Both statements are false.

   E. No answer can be given based on the information provided.

5. **(2 points)** CUDA streams enable parallel execution of GPU kernels. Imagine that two applications each need to execute three independent kernels. Based on the kernel timings below, which application, **A or B**, do you expect to gain the most performance from use of streams?

	Application A	**Application B**
Kernel X execution time	20 msec	5 msec
Kernel Y execution time	20 msec	20 msec
Kernel Z execution time	20 msec	35 msec

Problem 2 (27 points): Parallel Reduction and Prefix

Parts 1 and 2 pertain to the parallel reduction algorithm used in Lab 5.1.

1. **(12 points)** Which of the following statements is **TRUE**?
YOUR ANSWER SHOULD INCLUDE THE LETTER OF EVERY TRUE STATEMENT.

   A. The number of `__syncthreads` operations performed by each thread block depends on the size of the input array.

   B. The only requirement on the reduction operator is that it be associative.

   C. The algorithm performs fewer than BLOCK_SIZE reduction operations (for example, floating-point additions).

   D. On average, each thread accesses global memory approximately one time.

   E. Data in the input are logically commuted (reordered) to reduce branch divergence within CUDA warps.

   F. Each iteration of the reduction loop requires a `__syncthreads` call in order to prevent some threads from overwriting values (in shared memory) that other threads may not yet have read.

2. **(3 points)** What is the best approximation for the average number of distinct shared memory locations accessed by each thread during the reduction (that is, not counting reading data from global memory nor writing back the reduced value)? **PLEASE SELECT EXACTLY ONE ANSWER.**

   A. 1

   B. 2

   C. 3

   D. log2(BLOCK_SIZE)

   E. BLOCK_SIZE

   F. 2 × BLOCK_SIZE

Problem 2, continued:

The remaining parts pertain to floating-point addition scan kernels.

PLEASE SELECT EXACTLY ONE ANSWER FOR EACH QUESTION.

3. **(3 points)** Across all thread blocks in the grid, what is the MAXIMUM number of distinct warps that experience branch divergence during execution of a Kogge-Stone kernel with BLOCK_SIZE=1,024 threads in each thread block?

   A. 1 **D.** $\log_2$(BLOCK_SIZE)

   B. 3 **E.** BLOCK_SIZE / 64

   C. BLOCK_SIZE / 32 **F.** 2

4. **(3 points)** Let's say that a particular GPU can execute 10 floating-point operations for each global memory access before computation becomes the bottleneck. As you know, a Kogge-Stone kernel uses more computation than does a Brent-Kung kernel. At what number of threads per block does computation start to slow down the Kogge-Stone kernel on the given GPU?

   A. 2^{20} **D.** 400

   B. 256 **E.** 2^{10}

   C. 20 **F.** 10

5. **(3 points)** A Brent-Kung kernel is executed with 2,048 elements in each thread block's shared memory array. Which of the following best approximates the number of floating-point add operations performed per thread block in the reduction and post scan steps (summed together)?

   A. 2048×1024 **D.** $2 \times (2048 - 1)$

   B. $1024 \times \log_2(1024)$ **E.** 2046

   C. 2048 **F.** $2048 \times \log_2(2048)$

6. **(3 points)** Suppose that we need to perform a scan on an input of 2^{40} elements. Our GPU supports at most 2^{10} threads per block and at most 2^{11} blocks per grid. Using Brent-Kung in a hierarchical fashion, with none of the scan work done by the host, how many times do we need to launch the Brent-Kung kernel? (Reminder: $2^A/2^B = 2^{(A-B)}$.)

   A. $2^{18} + 3$ **D.** $2^{19} + 2^9 + 1$

   B. $2^{19} + 1$ **E.** $2^{29} + 2^{18} + 2^7 + 1$

   C. 3 **F.** $2^{18} + 2^7 + 1$

Problem 3 (26 points): Fun with Histograms

1. **(12 points)** Fill in the blanks to complete the histogram kernel function below with privatization.
   USE ONLY ONE LINE PER BLANK IN YOUR ANSWERS.

```
__global__ void histogram_kernel (unsigned char *Input,
                                   int *Output, unsigned int num) {

  __shared__ int histogram[256];

  int index = blockIdx.x * blockDim.x + threadIdx.x;

  if(threadIdx.x < 256) {
    histogram[index] = 0;
  }
  __syncthreads();

  int i;
  int stride = /* ANSWER A GOES HERE. */ ;

  for(i = index; i < num; i += stride) {
    /* ANSWER B GOES HERE. */ ;

    /* LOCATION FOR PART 2 OF THE PROBLEM. */
  }

  __syncthreads();

  if( /* ANSWER C GOES HERE. */ ) {
    /* ANSWER D GOES HERE. */ ;
  }

}
```

2. **(14 points)** Which of the following statements is **TRUE**?
   YOUR ANSWER SHOULD INCLUDE THE LETTER OF EVERY TRUE STATEMENT.

   A. For correct execution of the kernel in **Part 1**, a call to `__syncthreads` must be added at the location marked in the code.

   B. Privatization improves performance by transforming global atomic operations into atomic operations on shared memory locations.

   C. To reduce atomic conflicts on shared memory, we can make four copies of the histogram for each thread block and have one quarter of the threads access each copy. Doing so does NOT imply increasing the number of global atomic operations needed.

   D. Adding a call to `__syncthreads` at the location marked in the code for **Part 1** is not safe in CUDA.

   E. Sorting the input makes the kernel in **Part 1** execute more slowly.

   F. If the blanks are filled as necessary and any necessary calls to `__syncthreads` are added, the kernel in **Part 1** works correctly for any number of threads per thread block (blockDim.x) between 32 and 1024.

   G. To reduce atomic conflicts on shared memory, we can make two copies of the histogram for each thread block and have half of the threads access each copy. Doing so, however, implies introducing branch divergence.

Problem 4 (24 points): Sparing the Zeroes

In class, we learned about different representations of sparse matrices. **Parts** 1 **through 5** pertain to the sparse matrix shown below.

FOR EACH QUESTION, YOUR ANSWER SHOULD INCLUDE THE LETTER OF ALL CORRECT ANSWERS.

$$\begin{bmatrix} 2 & 9 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 6 & 4 \\ 10 & 0 & 7 & 0 & 0 \\ 0 & 0 & 8 & 5 & 0 \end{bmatrix}$$

Answers for **Parts 1 through 5** are as follows:

A. CSR
B. ELL
C. COO
D. JDS
E. JDS-Transposed
F. none of the above

1. **(3 points)** Which sparse matrix formats can have the following data layout for the array of non-zero matrix values in memory?

3	6	4	2	9	10	7	8	5	1

2. **(3 points)** Which sparse matrix formats can have the following data layout for the array of non-zero matrix values in memory?

2	9	1	3	6	4	10	7	8	5

3. **(3 points)** Which sparse matrix formats can have the following data layout for the array of non-zero matrix values in memory?

3	2	10	8	1	6	9	7	5	4

4. **(3 points)** Which sparse matrix formats can have the following data layout for the array of non-zero matrix values in memory?

2	1	0	3	10	8	9	0	0	6	7	5	0	0	0	4	0	0

5. **(3 points)** Which sparse matrix formats can have the following data layout for the array of non-zero matrix values in memory?

4	1	9	3	7	8	2	6	10	5

Problem 4, continued:

PLEASE SELECT EXACTLY ONE ANSWER FOR EACH QUESTION BELOW.

6. **(3 points)** Consider a matrix of 100 rows, 423 columns, and a total of 1,000 non-zero values. The maximum number of non-zero elements in any row is 50. If we pack the matrix into the ELL format, how many additional zeroes are needed for padding?

A. (423 – 50) × 100

B. (100 × 423) – 1000

C. (100 – 50) × 423

D. (100 × 50) – 1000

E. (50 – 42) × 100

F. insufficient information has been provided to answer the question

7. **(3 points)** The sparse matrix-vector multiply kernel below executes on a sparse matrix with 100 rows and 64 columns, including four empty rows (all zeroes). Each of the matrix's non-empty rows has 10 non-zero values, and the matrix is encoded using the ELL format.

```
__global__ void SpMV_ELL(int num_rows, float *data, int *col_index,
                         int num_elem, float *x, float *y) {
   int row = blockIdx.x * blockDim.x + threadIdx.x;
   if (row < num_rows) {
         float dot = 0;
         for (int i = 0; i < num_elem; i++) {
               dot += data[row+i*num_rows]*x[col_index[row+i*num_rows]];
         }
         y[row] = dot;
   }
}
```

How many warps must be launched to execute the kernel on the given matrix? Do any of those warps experience branch divergence?

A. After the transpose, the resulting matrix has 64 rows, so we need two warps, and there is no branch divergence.

B. Four warps are needed to cover the 100 rows, and all warps may experience branch divergence due to the indirect indexing used when accessing values from the vector x.

C. Thirty warps (960 threads) are needed to cover the 960 non-zero values in the matrix, and one warp experiences branch divergence—the one that crosses the row count boundary in the `if` statement.

D. Three warps are needed to cover the 96 non-empty rows, and there is no branch divergence.

E. Four warps are needed to cover the 100 rows. The last warp experiences branch divergence, as only some of its threads are needed for computation.

F. Three warps are needed to cover the 96 non-empty rows, and all warps may experience branch divergence due to the indirect indexing used when accessing values from the vector x.

Problem 4, continued:

PLEASE SELECT EXACTLY ONE ANSWER FOR THE QUESTION BELOW.

8. **(3 points)** The sparse matrix-vector multiply kernel below executes on a sparse matrix with 100 rows and 64 columns, including four empty rows (all zeroes). Each of the matrix's non-empty rows has 10 non-zero values. The matrix is encoded using the JDS-Transposed format, and **the four empty rows are simply left out of the encoding**, so `num_rows` is 96.

```
__global__ void SpMV_JDS_T(int num_rows, float *data, int *col_index,
                           int *jds_t_col_ptr, int *jds_row_perm, float *x,
                           float *y) {
   int row = blockIdx.x * blockDim.x + threadIdx.x;
   if (row < num_rows) {
      float dot = 0;
      unsigned int sec = 0;
      while (jds_t_col_ptr[sec+1]-jds_t_col_ptr[sec] > row){
         dot += (data[jds_t_col_ptr[sec]+row] *
                 x[col_index[jds_t_col_ptr[sec]+row]]);
         sec++;
      }
      y[jds_row_perm[row]] = dot;
   }
}
```

How many warps must be launched to execute the kernel on the given matrix? Do any of those warps experience branch divergence? Is the result correct, given that the encoding is missing the empty rows?

A. Three warps are needed to cover the 96 non-empty rows, and there is no branch divergence. The result cannot be correct without inclusion of the empty rows, however.

B. After the transpose, the resulting matrix has 64 rows, so we need two warps, and there is no branch divergence. The result is correct if y is initialized to 0.

C. 96 warps are needed to cover the non-empty rows. Since each warp handles exactly 10 non-zero values, all warps experience branch divergence. The result is correct if y is initialized to 0.

D. Three warps are needed to cover the 96 non-empty rows, and all warps may experience branch divergence due to variations in the test for the `while` loop. The result cannot be correct without inclusion of the empty rows, however.

E. Three warps are needed to cover the 96 non-empty rows, and all warps may experience branch divergence due to variations in the test for the `while` loop. The result is correct if y is initialized to 0.

F. Three warps are needed to cover the 96 non-empty rows, and there is no branch divergence. The result is correct if y is initialized to 0.

Problem 5 (9 points): Guest Lecture Topics

PLEASE SELECT EXACTLY ONE ANSWER FOR EACH QUESTION.

1. **(3 points)** Prof. Lumetta has unwisely chosen his own name ("lumetta") as his password for a high-security computing system. According to Prof. Levchenko's calculations, how long would it take a high-performance GPU to use brute force to crack Prof. Lumetta's password, assuming that his password is alphanumeric (consisting of letters of either case and digits)?

   A. About a day (about 2^{16} seconds).

   B. Forever, effectively (more than 2^{30} seconds). Prof. Lumetta is not so silly.

   C. Roughly a month (about 2^{21} seconds).

   D. A little over half an hour (about 2^{11} seconds).

2. **(3 points)** How does Prof. El Hajj suggest reducing the number of kernels launched dynamically when executing on a GPU?

   A. Apply a threshold, only launching a child kernel when the child kernel has enough parallelism (more than the threshold).

   B. Limit recursive nesting depth of kernel launches: kernels that execute at the maximum depth do not launch child kernels.

   C. Avoid overwhelming the GPU by applying a threshold, only launching a child kernel when the child kernel requires few enough thread blocks.

   D. Reduce the number of threads used per thread block in each child kernel so that more thread blocks can execute simultaneously on each SM in the GPU.

3. **(3 points)** According to Mr. Pearson, how many different metrics of kernel execution are supported by NVIDIA's Nsight tool?

   A. Around 1,000, although most of their meanings are fairly arcane and unlikely to be immediately useful to every GPU programmer.

   B. Exactly 32—one per thread in a warp.

   C. Around 100, including a variety of important architectural, microarchitectural, scheduling, and runtime elements.

   D. Roughly 10,000, or even more if one includes Nsight's ability to extend capabilities by executing user-defined code within the GPU hypervisor.