CSCI-UA.0480-051: Parallel Computing
Final Exam (May 12th, 2022)
Total: 100 points

Important Notes- READ BEFORE SOLVING THE EXAM

- **If you perceive any ambiguity in any of the questions, state your assumptions clearly and solve the problem based on your assumptions. We will grade both your solutions and your assumptions.**
- **This exam is take-home.**
- **The exam is posted on Brightspace, at 2pm EST of May 12th.**
- **You have up to 23 hours and 55 minutes from 2pm EST of May 12th to submit your answers on Brightspace (in the assignments section).**
- **You are allowed only one submission, unlike assignments and labs.**
- **Your answers must be very focused. You may be penalized for wrong answers and for putting irrelevant information in your answers.**
- **You must upload one pdf file.**
- **Your answer sheet must have a cover page (as indicated below) and one problem answer per page (e.g., problem 1 in separate page, problem 2 in another separate page, etc.).**
- **This exam has 4 problems totaling 100 points.**
- **The very first page of your answer is the cover page and must contain:**
    - **You Last Name**
    - **Your First Name**
    - **Your NetID**
    - **Copy and paste the honor code showed in the rectangle at the bottom of this page.**

Honor code (copy and paste the whole text shown below, in the rectangle, to the first page, cover page, of your exam)

Problem 1

a. [5] A programmer is developing a program for a company. The programmer has parallelized 90% of the program. The programmer has been informed that the problem size will remain the same no matter how many cores will be used. What is the expected speedup on 10 cores? Show your steps. Make any needed assumptions if any. Round the speedup to the nearest single digit after the floating point.

b. [5] A programmer was able to write a parallel program, running on four cores, and obtain a speedup of 2. What is the fraction of the sequential part of this program? Show all the steps to get full credit. Assume the cores are superscalar but not with hyperthreading capability.

c. [15] State three reasons as to why coherence protocol has a negative effect on performance in a shared-memory machine.

Problem 2

Suppose we have 4 processes. Each one has an array of int called A. Assume these are the only processes in a communicator called NEWCOMM. Each array consists of four int, and initialized as follows:

	Initial values in each process's array A			
Process 0	1	8	9	8
Process 1	2	7	10	15
Process 2	3	6	11	14
Process 3	4	5	12	13

a. [7] Each of the four processes has another array of four int called B. Array B in each process are not initialized. There is a function call that all processes made. The result of this call is that the content of arrays B changed to the following values. Write down that call with all its arguments. It is one line not more.

	Values in each process's array B after the call			
Process 0	0	0	8	8
Process 1	0	0	8	8
Process 2	0	0	8	8
Process 3	0	0	8	8

b. [3] What is the *minimum* number of communicators that each one of these four processes can be a member of? Justify.

c. [5] Suppose we execute the above processes on a machine with 16 cores, each one is 2-way hyperthreading. Is there any chance these four processes will fully use all the cores simultaneously? Explain.

Problem 3

a. [5] We know that statements: **#pragma omp single** and **#pragma omp master** both make its following statement be executed by only one thread. Give a scenario where we *must* use the **#pragma omp master** and not the other one.

b. [5] Given the following nested-loops, is it more beneficial to parallelize (using OpenMP), the outer loop? or the inner loop? And why?

```
for (i=1; i < n; i++) {
  for (j = 0; j < n; j++) {
    v[j][i] = (v[j][i-1] + v[j][i] + v[j][i+1])/4);
  }
}
```

c. [20] For each one of the following scenarios, state whether you will use atomic, critical, or locks in OpenMP to deal with critical sections. If there are several possible choices in a scenario, pick the one that gives the best performance. If there are several possible solutions for a scenario, each of which gives the same performance as the others, then write all of them. Be as specific as possible. No justifications are needed for your choices.

Scenario	What will you use?
Several critical sections, each one consists of several lines of code, and we know the number of critical sections at programming time.	
Several critical sections, each one consists of one line of code in the form of "variable++" (where variable is any variable name), and we know the number of critical sections at programming time.	
We don't know the exact number of critical sections at programming time, but we know them at execution time.	
Only one critical section that consists of several lines of code	

Problem 4

Suppose we have the following code snippet. The programmer launches the kernel with:
printing<<<3,2>>>( );

```
__global__ printing(){

        __shared__ int x = 7;
        if(blockIdx.x > 0)
                printf("%d", blockIdx.x);
        else
                printf("x");
        __synchthreads();
        printf("%d", threadIdx.x);
}
```

a. [12 points] For each one of the following statements, specify whether it is a "possible"
output or "not possible" output (i.e. can never be printed on the screen no matter how
many times we execute the kernel). No need to justify your choice, just write (a. , b. , …)
and next to each one ("possible", or "not possible").
a. 770100010001   b. 000100011101   c. 701020711121
d. 777000000111   e. 000000011177   f. 001077010001

b. [6 points] If we remove the __shared__ keyword, which ones of your answers above
will change? Justify.

c. [6 points] If we remove the __synchthreads(), which ones of your answers (in part a)
above will change? Justify.

d. [6 points] How many warps are created when the kernel is launched? Explain.