CSCI-UA.0480-051: Parallel Computing
Midterm Exam (Oct 19th, 2021)

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 the beginning of the Oct 19th lecture.
- You have up to 23 hours and 55 minutes from the beginning of the Oct 19th lecture to submit 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 a 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 6 problems totaling 100 points. The very first page of your answer is the cover page and must contain:
  - Your Last Name
  - Your First Name
  - Your NetID
  - Copy and paste the honor code shown below.

Honor code (copy and paste to the first page of your exam)

- You may use the textbook, slides, and any notes you have. But you may not use the internet.

- You may NOT use communication tools to collaborate with other humans. This includes but is not limited to G-Chat, Messenger, E-mail, etc.
- Do not try to search for answers on the internet it will show in your answer, and you will earn an immediate grade of 0.
- Anyone found sharing answers or communicating with another student during the exam period will earn an immediate grade of 0.
- "I understand the ground rules and agree to abide by them. I will not share answers or assist another student during this exam, nor will I seek assistance from another student or attempt to view their answers."

Problem 1

a. [**10**] Explain the difference between strong scaling and weak scaling in parallel computing. Give an example scenario where strong scaling would be preferred and another where weak scaling would be more appropriate.

b. [**10**] What is Amdahl's Law? Explain its implications for the speedup achievable through parallelization. Give a concrete example of a program where Amdahl's Law would severely limit the potential speedup.

c. [**10**] Describe the concept of a race condition in concurrent programming. Provide a simple code snippet (in any language) demonstrating a race condition, and explain how it could be avoided.

Problem 2

Consider a parallel program that performs matrix multiplication of two $n \times n$ matrices.

a. [**10**] Describe a simple parallel algorithm for matrix multiplication using a master-worker approach. Explain how the work is divided among the workers and how the results are aggregated.

b. [**10**] Analyze the runtime complexity of your algorithm in terms of n and the number of processors p. Identify any potential bottlenecks in your design.

c. [**10**] Discuss the impact of communication overhead on the performance of your parallel matrix multiplication algorithm. How could you potentially reduce this overhead?

Problem 3

a. [**15**] Describe the differences between OpenMP and MPI. When would you choose to use each paradigm? Provide specific application examples where one would be more suitable than the other.

Problem 4

Suppose we have the following DAG that represents different tasks and their dependencies:

(Insert a simple DAG here, e.g., A -¿ B, A -¿ C, B -¿ D, C -¿ D, D -¿ E)

The following table shows the execution time of each task in milliseconds:

Task	Execution Time (ms)
A	10
B	5
C	8
D	12
E	7

a. **[10]** Determine the minimum execution time of this DAG on a single processor.

b. **[10]** Assuming you have two processors, schedule the tasks to minimize the overall execution time. Show your schedule and calculate the resulting execution time.

Problem 5

Consider a shared-memory parallel program with two threads. Thread 1 increments a shared counter variable 1000 times, and Thread 2 decrements the same counter variable 1000 times.

a. **[10]** Explain why using a simple increment/decrement operation without any synchronization mechanisms might lead to an incorrect final value of the counter.

b. **[10]** Describe how you would use mutex locks (or other synchronization primitives) to ensure that the final value of the counter is correct. Provide a code snippet (pseudocode or a real language) illustrating your solution.

Problem 6

a. **[10]** Explain the concept of false sharing in a cache-coherent multiprocessor system. Provide an example scenario where false sharing could significantly impact performance.

b. **[10]** Discuss techniques to mitigate the negative effects of false sharing.