Parallel Computing Practice Exam

Name: _____
Date: _____

Section 1: Multiple Choice (1 point each)

1. Which of the following is NOT a primary challenge in parallel computing?
a) Load balancing
b) Data dependency
c) Increased processing speed
d) Communication overhead

Answer: _____

2. What is Amdahl's Law used to calculate?
a) The maximum speedup achievable by parallelizing a program
b) The optimal number of processors for a given problem
c) The communication overhead in a parallel system
d) The memory bandwidth of a parallel system

Answer: _____

Section 2: Short Answer (3 points each)

3. Briefly explain the difference between shared memory and distributed memory parallel architectures. Include at least one advantage and one disadvantage of each.

Answer:

4. Describe two common techniques used for handling race conditions in parallel programs.

Answer:

Section 3: Problem Solving (5 points)

5. Consider the following task: Sorting a list of 1,000,000 integers. Describe how you would approach this problem using a parallel algorithm. Specifically:

* What parallel programming paradigm would you choose (e.g., shared memory, message passing)? Justify your choice.
* What algorithm would you use for sorting? Explain why it is suitable for parallel processing.
* Briefly outline the steps involved in parallelizing the algorithm. Consider load balancing and data distribution.

Answer:

Answer Key:

1. c) Increased processing speed
2. a) The maximum speedup achievable by parallelizing a program
3. See expected answer below
4. See expected answer below
5. See expected answer below

Expected Answers:

3. Shared Memory vs. Distributed Memory:

* **Shared Memory:** Processors share a common address space. Advantage: Easier programming model, simpler data sharing. Disadvantage: Scalability limitations, potential for memory contention.
* **Distributed Memory:** Processors have their own private memory. Advantage: Better scalability, less memory contention. Disadvantage: More complex programming (message passing), more overhead for data communication.

4. Handling Race Conditions:

Two common techniques are:

* **Mutual Exclusion (Mutexes):** A lock that allows only one thread to access a shared resource at a time.
* **Semaphores:** A more generalized synchronization primitive that allows for controlling access to a resource based on a counter.

5. Parallelizing Sorting:

* **Paradigm:** Shared memory would be suitable for this problem, as it simplifies data sharing between threads. However, for extremely large datasets, a distributed memory approach might be necessary. The choice depends on the available resources.
* **Algorithm:** Merge Sort is a good choice for parallel sorting. It recursively divides the list into smaller sublists that can be sorted independently. This inherent divide-and-conquer nature maps well to parallel processing. Other algorithms like Quicksort can also be parallelized, but their performance can be less predictable due to potential load imbalances.
* **Steps:**

1. Divide the list of integers into smaller sublists, one for each processor.
2. Each processor independently sorts its sublist using a sequential sorting algorithm (e.g., merge sort).
3. Merge the sorted sublists together using a parallel merge algorithm. This step may require communication between processors to exchange data.
4. Load balancing can be achieved by dynamically assigning sublists of roughly equal size to each processor.

This answer key provides a framework; variations in approach are possible and should be evaluated on their merits. The quality of the student's explanation and justification is key to awarding full points.