Questions

1. (although a proof will not appear on the final exam, this question is good nonetheless)

   Prove : ¬ (¬ P ∨ Q) ∨ Q ∧ ¬(R => Q) ∧ ¬ Q => P ∧ R

2. True/False Doubling the amount of threads being used will cut the execution time in half.

3. What does MPI stand for?

4. How many stages are required to implement a dissemination barrier for 12 threads?

5. Give an example of a section of code that has flow dependence and anti dependence.

6. True/False An instruction is control dependent on a preceding instruction if the output of the former can determine if the latter should be executed.

7. Eventual entry is what kind of property?

   A) Safety
   B) Liveness
   C) Both safety and liveness
   D) None of the above

8. True/False The state of a program is the names of all the variables within scope at a point in time.

9. Consider an undirected graph (network) with 5 vertices {A,B,C,D,E} and 5 edges. Each node represents a computer which has a single randomly assigned integer value. The goal is for all 5 vertices to know of the largest integer value among them. Self-looping edges (an edge emanating from and going to the same node) are not allowed, and two nodes cannot have more than one edge between them. If the nodes communicate via messages sent back-and-forth concurrently, then among which vertices should the 5 edges be placed to ensure worst performance? Assume the “cost” of sending a message from one node to another is 1. If n hops are required, then the cost incurred is n.
10. What is a race condition?

11. For an M x N matrix where M >> N, we have a task that sums the entries of this matrix. Which of the following statements is true

A. Matrix data items are placed into cache in columns
B. All columns of some rows, but not all columns of all rows fit into cache
C. If all the columns of each row are NOT stored in cache, there is a guaranteed cache hit
D. By summing the rows of each array, a cache hit is NOT guaranteed

12. Which of these choices is FALSE regarding disadvantage for asynchronous message passing?

A. Flooding of network, which might cause congestion
B. Communication overhead diminishes a program’s performance
C. Main process that is broadcasting has an excessively high computational burden
D. Method of making the connection is error prone

13. True/False You can create and utilize only one instance of a semaphore at a time.

14. What safety/liveness property does the tie-breaker algorithm address that could not be guaranteed by previous solutions to the critical section problem?

15. What is spatial locality (in reference to caching)?

16. What is the minimum number of operations it will take for a butterfly barrier to completely synchronize? Assume each process does await and assign twice per stage, and that that counts as one operation.

17. Why does executing n threads not result in a n times speed up?

18. True/False A Butterfly barrier is best used for an odd number of worker processes.
19. What of the following is not a shortcoming of a semaphore?

A. You (a programmer) must be careful NOT to omit an increment or decrement somewhere in your code, or to use the wrong semaphore (in case there are multiple ones being used).
B. Semaphores are global, thus you must examine the entire program to know how they are used.
C. Semaphores have getter methods which can retrieve the current value of the semaphore that leads to atomicity problems.
D. Semaphores provide BOTH mutual exclusion and synchronization techniques, but those concepts cannot be used independently.

20. True/False Throughput is negatively affected when balancing a pipeline.

21. What is the average access time of cache?

22. What is the diff between a thread and a process?

23. What problem does Test and Test and Set solve that Test and Set introduces? How does it solve it?

Answers

1. \[ \neg (\neg P \lor Q) \]
   \[ P \land \neg Q \]
   \[ \neg((R \Rightarrow Q) \land \neg Q) \]
   \[ R \land \neg Q \lor Q \]
   \[ P \land \neg Q \lor Q \land R \land \neg Q \lor Q \]
   \[ P \land \text{true} \land R \land \text{true} \]
   \[ P \land R \]

2. False. The overhead involved making a thread slows down the execution and any process not threaded.
3. Message Passing Interface
4. \( \lceil \log_2(12) \rceil = 4 \) stages.
5. \( x = y; \ y = x \)
6. False
7. B
8. False
9. There are many ways to send messages such that it takes 3 units of time. The requirement here is that there are nodes with a distance of greater than 2 from some node (longest shortest path from A to ?). Note a circular link doesn't work since we can send messages to everybody by passing both ways, and given messages can be broadcast concurrently we could send left/right in 2 units of time. One such topology includes the following Edge set: \( E=\{\{A,C\},\{B,C\},\{A,B\},\{D,C\},\{E,D\}\} \).
10. A race day condition is a special condition which may occur inside of a critical section, in which the different possible histories results in different outputs.

11. B

12. D

13. False

14. Fairness / A process eventually enters critical section

15. Spatial locality is the idea that if a cached memory location is used, then it is likely that a nearby memory location will be used in the near future.

16. A butterfly barrier only works with the number of processes that are a multiple of two. To get the operations is take the number of process / 2 and multiply it by log_2 (number of processes). For 8 processes there are 3 stages (log_2 8). The number of operations per stage is 4. 4 * 3 = 12

17. Because the process of creating threads causes overhead time that accounts to the total processing time.

18. False (all powers of 2 are NOT odd)

19. C

20. False

21. hit time + miss rate * miss penalty

22. A process can consist of more than one thread. A thread is the smallest part of a process that can run concurrently with other threads of the process. Threads are often referred to as lightweight processes as it uses a process’ address space and shares it with other threads in the process.

23. Test and Set has every process checking and changing the lock to true even if the lock is already true. This leads to many processes constantly and unnecessarily writing to the same variable in memory. TTS introduces spin loops that keep processes from writing to the lock variable while the lock is set to true.