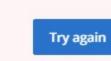
Module 1 Quiz
Graded Quiz • 30 min

! Try again once you are ready
TO PASS 70% or higher



GRADE 62.5%

## **Module 1 Quiz**

62.5%

1. Which of the following primitive operations on threads were introduced in Lecture 1.1?

Please choose all options that are correct.

0.75 / 1 point

Incorrect

2. Which of the following statements are true, based on what you learned about threads in Lecture 1.1?

0.5 / 1 point

Please choose all options that are correct.

Incorrect

Please choose all options that are correct.

Incorrect

4. Consider a parallel program with two structured locks, L1 and L2, (as introduced in Lecture 1.2), and

1/1 point

100 threads that are started in parallel such that each of them executes the following snippet of code:

1 \* synchronized(L1) {
2 \* synchronized(L2) {
3 S1
4 }
5 }

The statement "This program is guaranteed to be free from deadlocks" is:

Which of the following are true regarding unstructured read-write locks introduced in Lecture 1.3?

0.667 / 1 point

Please choose all options that are correct.

✓ Correct

6. Which of the following best describe the differences between structured and unstructured locks introduced in Lectures 1.2 and 1.3?

0.667 / 1 point

Please choose all options that are correct.

Incorrect

concurrently, as possibly leading to deadlock, livelock or neither. Recall that deadlock and livelock were discussed in Lecture 1.4.

7. Please classify the following five scenarios, each employing two threads (T1 and T2) running

0 / 1 point

Scenario 1:

Scenario 2: A and B are declared and initialized as distinct objects

```
1 // Thread T1
2 * {
3 * synchronized(A) {
4 * synchronized(B) {
5 . . . // sequential code
}
```

Scenario 3: A and B are declared and initialized as distinct objects

1 // Thread T1

Scenario 4: x is declared and initialized as a shared Integer object

1 // Thread T1

Scenario 5: x is declared and initialized as a shared Integer object

1 // Thread T1

What is the difference between deadlock and livelock? (Recall that deadlock and livelock were

Incorrect

discussed in Lecture 1.4.)

1/1 point

0 / 1 point

Lecture 1.5. In this problem, you will evaluate one such algorithm.

The following simple algorithm can deadlock with all philosophers holding their left chopsticks.

9. Many algorithms have been proposed to address the Dining Philosophers problem introduced in

the chopsticks in the opposite order. More precisely, we propose separate algorithms for the evennumbered and odd-numbered philosophers. Assume the philosophers are numbered 1, ..., n clockwise around the table.

```
2 - while (True) {
 3 acquire philosopher's left chopstick
 4 acquire philosopher's right chopstick
 5 eat
      release right chopstick
      release left chopstick
 10 // All odd-numbered philosophers:
 11 → while (True) {
 12 acquire philosopher's right chopstick
 13 acquire philosopher's left chopstick
 14
 15
      release left chopstick
 16
      release right chopstick
17 }
```

What liveness issues could arise if we have an even number of philosophers? What about if we had

Incorrect

1 // All even-numbered philosophers:

an odd number of philosophers?

10. Towards the end of Lecture 1.5, Dr. Sarkar mentioned that deadlock and livelock can be avoided in the Dining Philosophers problem by modifying the "all acquire left fork first" algorithm such that n-1 philosophers attempt to acquire their left fork first, and 1 philosopher attempts to acquire its right fork first. However, nothing was mentioned about the impact of this modification on another important liveness issue: starvation. How could starvation occur, or not occur, with this modification presented at the end of Lecture 1.5? 1/1 point