Module 4 Quiz

Quiz, 10 questions

1 point

1.
What is a potential downside of the shown getAndAdd() function, which employs optimistic concurrency in the following code block?

```
1  getAndAdd(int delta) {
2    while (true) {
3      cur = this.get();
4      next = cur + delta;
5      if (this.compareAndSet(cur, next) return cur;
6    }
7  }
```

- A. It may perform many unused computations.
- B. It requires the use of expensive locks.
- C. It may deadlock.
- D. It may livelock.

---

1 point

2.
Under what circumstances might optimistic concurrency be a good strategy when designing a concurrent algorithm?

- A. Computation on the shared object is very expensive compared to the overhead of locks.
- B. You expect very low contention.
- C. The optimistically computed operation has side effects.

---

1 point

Module 4 Quiz

Quiz, 10 questions

3.

Which variables in an implementation of a sequential queue would need to be handled differently in a concurrent implementation?

- A. Since not all variables would be part of a data race, only those that would be part of a data race.
- B. Since all variables would be part of a data race, all variables.
- C. All variables that are used in an enqueue operation.
- D. All variables that are used in a dequeue operation.
- E. None of the above.

1 point

4.

What's the best way to modify TAIL.NEXT in a concurrent implementation of ENQ(X)?

- A. LOCK(X) { TAIL.NEXT = X }
- B. ISOLATED(X) {TAIL.NEXT = X }
- C. TAIL.NEXT.COMPAREANDSET(TAIL, X)
- D. TAIL.NEXT.COMPAREANDSET(NULL, X)
- E. Both A and B

1 point

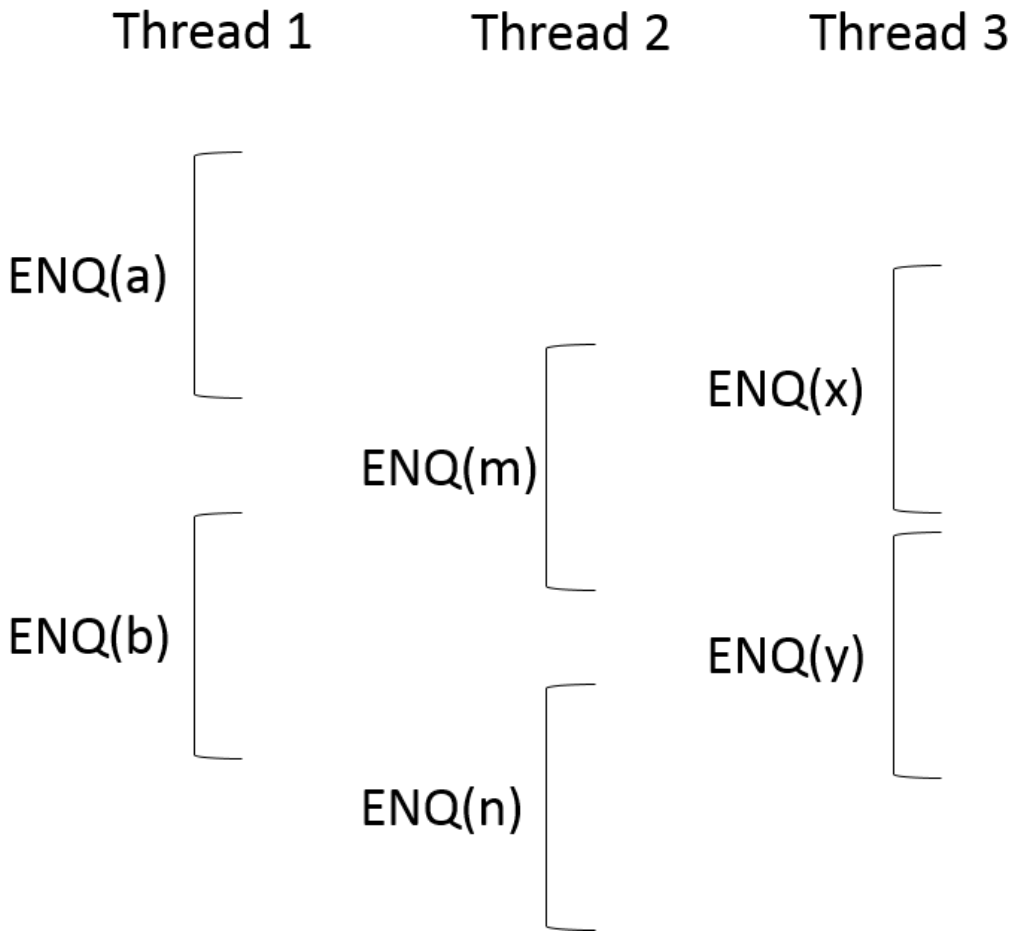
5.

Module 4 Quiz

Quiz, 10 questions

Consider the following timeline. Assuming the enqueue operations are linearizable, which of the below are possible results from a sequence of dequeue operations?

Please choose all options that are correct.



- [ ] A. a, m, x, b, n, y
- [ ] B. m, x, a, n, b, y
- [ ] C. m, x, y, a, b, n
- [ ] D. x, a, b, m, n, y
- [ ] E. x, a, y, m, n, b
- [ ] F. x, a, y, b, m, n

1 point

6.

Consider the scenario where threads T1 and T2 (and no other threads) are attempting to obtain a lock on L1. Which of the following are linearizable orderings of statements executed?

Please choose all options that are correct.

- [ ] A. 1) T1 calls L1.lock(). 2) T2 calls L1.lock(). 3) T2 is unable to obtain lock. 4) T1 successfully obtains lock.
- [ ] B. 1) T1 calls L1.lock(). 2) T2 calls L1.lock(). 3) T1 is unable to obtain lock. 4) T2 successfully obtains lock.
- [x] C. 1) T1 calls L1.lock(). 2) T2 calls L1.lock(). 3) T1 successfully obtains lock. 4) T2 is unable to obtain lock.
- [x] D. 1) T1 calls L1.lock(). 2) T2 calls L1.lock(). 3) T2 successfully obtains lock. 4) T1 is unable to obtain lock.
- [ ] E) 1) T1 calls L1.lock(). 2) T2 calls L1.lock(). 3) T2 successfully obtains lock. 4) T1 successfully obtains lock.
- [ ] F. 1) T1 calls L1.lock(). 2) T2 calls L1.lock(). 3) T1 is unable to obtain lock. 4) T2 is unable to obtain lock.

---

1 point

7.
If multiple threads are trying to write a value in the map for the same key using PUTIFABSENT(key, value), only one of the threads will succeed.

- (●) True
- ( ) False

---

1 point

8.
Which of the following operations are **not** linearizable?

Please choose all options that are correct.

- [ ] A. PUT(key, value)
- [ ] B. PUTIFABSENT(key, value)

- [x] C. PUTALL()
- [ ] D. CLEAR()
- [ ] E. GET (key)

1 point

9.
What is a possible downside of using locks to transform a sequential MST algorithm into a concurrent algorithm?

- (•) A. As our merged tree gets smaller, we have more collisions when using trylock, thus reducing performance.
- ( ) B. This method may not account for all data races.
- ( ) C. It results in a deadlock.
- ( ) D. The merging step could result in two processes attempting to merge the same nodes and thus result in a bug.
- ( ) E. There would be more code, which would hurt our brains.

1 point

10.
Is it possible for a minimum spanning tree to have the same total weight as its original graph?

- (•) A. Yes
- ( ) B. No

- [x] I, **Fan Yang**, understand that submitting work that isn't my own may result in permanent failure of this course or deactivation of my Coursera account.

Learn more about Coursera's Honor Code

Submit Quiz