New York University

CSCI-UA.0202-003: Operating Systems (Undergrad): Spring 2025

Quiz 3

- Write your full name on both:
  - the bubble sheet in the "Name" field
  - the quiz booklet
- Write your NYU NetID on the quiz booklet and the bubble sheet in the "ID" field
- Use a #2 pencil to fill in your answers on the bubble sheet
- This quiz contains 6 questions only. Each question has choices from A to D
- Fill the bubbles completely by darkening the entire circle, as shown in the example
- Only mark answers for questions 1-6. Do not mark any bubbles beyond question 6
- Choose only one answer per question
- Submit your bubble sheet together with your exam booklet

Name:

NetId:

1. In the monitor pattern we studied in the class:
   (a) Condition variables can be used without associated mutexes
   (b) Multiple threads can be in the critical section simultaneously
   (c) All method calls must be protected by the same mutex
   (d) Each method should use its own independent mutex

2. Which condition is NOT necessary for deadlock to occur?
   (a) Mutual exclusion
   (b) Hold and wait
   (c) Priority inversion
   (d) Circular wait

3. Given the following spinlock implementation:

```
void acquire(Spinlock *lock) {
    pushcli();
    while (xchg_val(&lock->locked, 1) == 1) {
        while (lock->locked) ;
    }
}
```

   What is the **main** purpose of the inner while loop?
   (a) It prevents thread starvation
   (b) It reduces the number of `xchg` instructions
   (c) It ensures that we are always trying to acquire the lock
   (d) It ensures mutual exclusion

4. What is the reading "An Investigation of the Therac-25 Accidents" about?
   (a) This is the official investigation report after the Therac-25 accidents
   (b) A detailed academic analysis of the accidents compiled from publicly available documents
   (c) A technical manual about the Therac-25's operation and maintenance
   (d) A legal document summarizing the lawsuits related to the accidents

5. Consider the following code snippet (assuming sequential consistency):

```
void transfer(Account* from, Account* to, int amount) {
    if (from->id < to->id) {
        mutex_lock(&from->mutex);
        mutex_lock(&to->mutex);
    } else {
        mutex_lock(&to->mutex);
        mutex_lock(&from->mutex);
    }

    from->balance -= amount;
    to->balance += amount;

    mutex_unlock(&to->mutex);
    mutex_unlock(&from->mutex);
}
```

Can this code lead to deadlock?
(a) Yes, because it locks multiple mutexes
(b) No, because mutexes are always acquired in a fixed order based on account ID
(c) Yes, because mutex release order differs from acquisition order
(d) No, but it can cause data race conditions

6. Which one of the following is **not** a contributing factor to the accident?
(a) Insufficient software testing and documentation
(b) Over-reliance on software controls instead of hardware interlocks
(c) Lack of proper instructions (with detailed error explanation) to the operators
(d) Did not reuse the software from previous generations of the machine