New York University

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

Quiz 2

- 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. Consider the following code snippet running in two concurrent threads (assuming sequential consistency):

```
// Initially x = 1, y = 0 (both are global variables)

// Thread 1                    // Thread 2
y = x;                         x = 2;
x = y + 1;                     y = x + 1;
```

   What values **could** x have after both threads complete?
   (a) Only 1 or 2
   (b) Only 2 or 3
   (c) 1, 2, or 3
   (d) Only 2

***Correction**: the correct answer should be 2, 3, and 4. Full point is given to those who chose (b).**

1. Which statement about condition variables is **FALSE**?
   (a) They must always be used with a mutex
   (b) They can be used to coordinate thread scheduling
   (c) The `wait()` operation atomically releases the associated mutex
   (d) They store the boolean condition being checked by threads

2. A common issue with using busy-waiting in the producer-consumer problem is:
   (a) It can lead to deadlocks between producer and consumer
   (b) It wastes CPU cycles while checking conditions repeatedly
   (c) It fails to maintain mutual exclusion
   (d) It causes buffer overflow

3. Which component is stored in a Process Control Block (PCB) but NOT in a Thread Control Block (TCB)?
   (a) Program counter value
   (b) Address space information

(c) Stack pointer
(d) Register values

4. Which property of concurrency solutions states that "if multiple threads attempt to enter their critical sections, at least one must eventually succeed"?
   (a) Mutual Exclusion
   (b) Progress
   (c) Bounded Waiting
   (d) Fairness

5. Consider this code using monitors with explicit locking:

```
class BoundedBuffer {
    mutex_t mutex;
    cond_t not_full;
    Buffer buffer;
    void put(Item item) {
        while (buffer.isFull()) {
            cond_wait(&not_full);
        }
        acquire(&mutex);
        buffer.add(item);
        release(&mutex);
        cond_signal(&not_full);
    }
}
```

   What monitor principle is violated?
   (a) The condition variable wait should be inside the critical section
   (b) The signal should occur before releasing the mutex
   (c) The mutex should be acquired before checking the condition
   (d) The condition check doesn't need to be in a while loop

***Correction**: (b) and (c) are both violated. Full point is given as long as you chose one of them.*