

Quiz2 Answer Key (Set-A)

[CS3.301 Operating Systems and Networks]

Question 1

Answer:

a)

a)

Initialization: [1 Mark for correct initialization]

```
semaphore mutex = 1 // to ensure mutual exclusion when  
                    // accessing the buffer
```

```
semaphore empty = N // counts empty slots          in the buffer
```

```
semaphore full = 0 // counts filled slots in the buffer
```

Producer: [1 Mark for correct Producer Code]

```
do {
```

```
    // produce an item
```

```
    item = produce_item();
```

```
    wait(empty); // decrement empty count (wait if no empty  
                // slot)
```

```
    wait(mutex); // enter critical section
```

```
    insert_item(item); // add item to buffer
```

```
    signal(mutex); // exit critical section
```

```
    signal(full); // increment full count (one more item available)
```

```
} while (true);
```

Consumer: [1 Mark for correct consumer pseudocode]

```
do {
```

```
    wait(full); // wait if buffer is empty
```

```
    wait(mutex); // enter critical section
```

```

    item = remove_item(); // remove item from buffer

    signal(mutex); // exit critical section
    signal(empty); // increment empty count (one more slot
available)

    consume_item(item);
} while (true);

```

b)

If both full and empty are initialized to 1:

The semaphores no longer represent the true buffer state.

- empty = 1 makes producers think there is only one empty slot, even if the buffer has N slots, so only one item can be produced before producers block. [0.5 Mark]
- full = 1 makes consumers think there is already one item available, a consumer may try to consume from an empty buffer, causing **underflow**. [0.5 Mark]

The buffer behaves incorrectly, allowing possible underflow or overflow.

If full is initialized to N:

The system assumes the buffer is already full.

- Consumers start consuming before any item is produced (underflow). [1 Mark]

Question 2

Answer:

Chooses any other option than (Favouring readers increases throughput but may starve writers) - Straight 0

Correctly selects option identifies that (Favoring readers increases throughput but may starve writers) - 1 Mark

Explains that allowing multiple readers to access shared data concurrently improves throughput compared to writers who require **exclusive access** - 1 Mark

Explains that favoring readers can cause writer starvation, as **continuous reader activity may indefinitely delay writers** - 1 Mark

If the student mentions about a greater number of readers than writers - 1 Mark

Question 3

Answer:

Chooses any other option than (The upper bound is $2T$) - Straight 0

Student correctly identifies that the appropriate upper bound is $2T$ (the time for two context switches). - 1 Mark

Explanation includes that a context switch involves putting a thread to sleep and waking it up, costing roughly $2T$ total, so spinning longer than this wastes CPU time. - 1 Mark

States that for longer critical sections (expected hold time $> 2T$), a mutex (blocking lock) is preferred to avoid busy waiting. - 1 Mark

Question 4

Answer:

Student selects **NS record** as the response type from the TLD server. - 1 Mark

For rest of the Options – Straight 0

Student correctly identifies that the TLD server does not provide the final IP address but rather **points to the authoritative name servers** for the next level domain (e.g., iiit.ac.in). - 2 Mark

Question 5

Answer: A, C, D

Explanation: After the host obtains its IP address, subnet mask, and default gateway via DHCP, it uses ARP to find the MAC address of the default gateway (necessary for communication outside the host's local subnet). Since the destination IP is outside the host's local subnet, the host will send the packet to the default gateway. The default gateway will then forward the packet based on the routing configuration.

Grading: 2 marks for correct option (no partial marks whatsoever) and 1 mark for reasoning

Question 6

Answer: B

Explanation: Each cache hit acts as a “reader” and the cache miss as “writer”, allowing increased throughput, concurrency and correctness.

A => Serialises access with no concurrency

C => Network optimization and not a concurrency mechanism for cache access

D => Waste a lot of CPU resources, preferred for low-contention use-case

Grading: 1.5marks for correct option and 1.5 for reasoning

Bonus Question

Answer:

When a philosopher makes a request for a chopstick, the request can be granted under the following conditions:

1. The philosopher already has two chopsticks, and there is at least one chopstick remaining.
2. The philosopher already has one chopstick, and there are at least two chopsticks remaining.

3. There is at least one chopstick remaining, and there is at least one philosopher who already has three chopsticks.
4. The philosopher has no chopsticks, there are two chopsticks remaining, and there is at least one other philosopher who already has two chopsticks

The rules correctly describe when a philosopher's request for a chopstick can be granted, ensuring logical and safe allocation.

- Full (1.5): All rules are correct and logically consistent.
- Partial (1.0): Minor logical error or ambiguity in one rule.
- Minimal (0.5): Multiple inaccuracies but shows partial understanding.
- None (0): Rules incorrect or irrelevant.

The answer covers all relevant cases and presents them clearly.

- Full (1.5): All cases covered clearly and fully.
- Partial (1.0): One case missing or unclear.
- Minimal (0.5): Multiple omissions or vague statements.
- None (0): Major omissions; incomplete answer.