Solutions

2.1

- Both reads see 10; the last write is 11 – final 11 ✓.

2.2

- Mutual exclusion, progress, bounded waiting ✓.

2.3

- The waiter sleeps in the OS queue instead of spinning – a spinning thread on the only core blocks the lock holder itself ✓.

3.1

- Maximum 6: all six increments land in order, each read sees the prior write ✓.
- Minimum 1: all six reads happen before any write (all read 0), then the six writes each store 1 ✓.
- Distinct final values: 6 – every value 1 to 6 is reachable ✓.
- Fix: wrap the increment in a lock (mutual exclusion) – the result is always 6 ✓.

3.2

- A holds L1, waits for L2; B holds L2, waits for L1: each waits forever for the other – circular wait across two holders ✓.
- Rule: always acquire locks in one global order (L1 before L2) ✓.
- Under the rule: B, holding nothing yet, takes L1 first – both threads queue on L1 and proceed one at a time ✓.
- Alternative: break hold-and-wait – request all locks at once (acquire-or-release) so no thread holds while waiting ✓.