CERTLABZ CERTIFICATION BOARD

Certified Quorum Developer (CQRD)

Ten Sample Questions

These ten questions are written at the difficulty of the live exam. Six carry a figure, a data table, a configuration or a code listing; four are plain scenario questions, as on the live paper. Every one is a scenario a working Quorum developer would meet, and none of them appears in the live question bank.

The live exam is multistage adaptive: how you score on one stage selects the difficulty of the next, so no two candidates sit the same paper. It is a performance-based exam: each form can carry up to 5 to 12 performance-based questions alongside the multiple-choice questions, which means you build, sequence or complete something rather than only choosing from a list. Answers and full explanations follow each question here; the live exam does not disclose item-level answers.

Question 1

Domain 1: Quorum and Enterprise Ethereum Foundations

DATA

Emma Wilson compares a public chain with Quorum, with one Quorum trait missing.

Feature	Public chain	Quorum
Native currency	Required	Optional
Membership	Open	(to choose)
Transactions	All public	Public or private
Consensus	Proof of stake	QBFT or Raft

Table 1.1 Public chain against Quorum.

What is Quorum membership?

A. Open to anyone
B. Fully anonymous
C. Permissioned
D. Unrestricted access

Correct answer: C

Quorum runs as a permissioned network where an operator controls which participants may join, unlike an open public chain. That closed membership is what lets enterprises trust the validator set and use private transactions.

Question 2 Domain 2: Consensus and Networking DATA

James Carter contrasts the two consensus options, with one best-fit cell missing.

Trait	QBFT	Raft
Fault model	Byzantine	Crash only
Finality	Immediate	Immediate
Best fit	(to choose)	Trusted nodes
Voting	Validators	Leader based

Table 2.1 QBFT against Raft.

What does QBFT best fit?

A. A single trusted host
B. Adversarial nodes
C. Only two nodes
D. No validators at all

Correct answer: B
QBFT tolerates Byzantine faults, so it suits networks where some validators may act maliciously or be less trusted. Raft only survives crash faults and assumes cooperative nodes, which limits it to fully trusted settings.

Question 3 Domain 3: Privacy and Private Transactions CONFIG

Olivia Davis reviews a private transaction that targets one peer.

```
const tx = {
  from: sender,
  to: contractAddr,
  data: payload,
  privateFor: [peerKey]
};
```

Listing 3.1 A private transaction.

What does privateFor set?

A. Who can read it
B. The gas price paid
C. Order of the nonces
D. The number of retries

Correct answer: A
The privateFor field lists the public keys of the parties allowed to receive and read the private payload, which the private transaction manager delivers off chain. Nodes outside that set see only a hash, so they cannot read the data or the resulting private state.

Question 4

Domain 4: Smart Contract Development

CONFIG

Liam Anderson inspects a guard inside a deployed contract.

```
pragma solidity ^0.8.0;
contract Vault {
  address owner;
  function drain() external {
    require(msg.sender == owner);
  }
}
```

Listing 4.1 A contract guard.

What does the require line enforce?

A. A payable call
B. Event logging on
C. Automatic upgrades
D. Owner only access

Correct answer: D
The require check reverts unless the caller equals the stored owner, so only the owner can run drain. This is a basic access control pattern; it does nothing for payments, event logging, or upgrades, which need separate code.

Question 5

Domain 5: Permissioning and Node Management

CONFIG

Ava Thompson reviews the permissioning config that lists nodes and account roles.

```
nodes:
  - enode://abc@10.0.0.1
  - enode://def@10.0.0.2
accounts:
  - 0xF1 role: admin
  - 0xA2 role: member
```

Listing 5.1 Permissioning config.

What does node permissioning restrict?

A. The token supply cap
B. The block gas limit
C. Which nodes connect

D The Solidity version

Correct answer: C
Node permissioning lists the enode identities allowed to peer with the network, so it controls which nodes may connect and take part. It has no bearing on gas limits, contract language versions, or any token supply.

Question 6

Domain 6: Security, Monitoring and Deployment

DATA

David Brown maps each risk to a mitigation, with one mitigation missing.

Risk	Mitigation
Reentrancy	Checks then effects
Unchecked call	(to choose)
Key exposure	Hardware modules
Node downtime	Redundant nodes

Table 6.1 Risks and mitigations.

How do you handle an unchecked call?

A Raise the gas price
B Check the result
C Disable all events
D Skip unit testing

Correct answer: B
A low level external call returns a success flag that must be checked, since it does not revert on failure by itself. Verifying the returned result catches silent failures, while gas price, events, and testing do not address the ignored return value.

Question 7

Domain 2: Consensus and Networking

A private network of well known banks wants immediate finality and cannot tolerate one malicious validator forging blocks. Which consensus fits best?

A Proof of work
B Raft consensus
C Proof of stake
D QBFT consensus

Correct answer: D
QBFT gives immediate finality and tolerates Byzantine validators, so a single malicious node cannot forge accepted blocks. Raft only handles crash faults, and the proof based schemes suit open public chains rather than a closed bank consortium.

Question 8 Domain 3: Privacy and Private Transactions

A team marks a transaction private to two parties, then is surprised a third node cannot see the resulting contract state. What best explains this?

A. State stays scoped
B. The node went offline
C. The chain forked hard
D. Gas ran out midway

Correct answer: A
Private state is scoped to the parties named in the private transaction, so nodes outside that set never receive the payload and cannot compute the same state. This is expected privacy behavior, not a fork, an outage, or an out of gas failure.

Question 9 Domain 4: Smart Contract Development

A contract has a bug, but the team deployed it as immutable with no upgrade path. Which pattern would have let them fix the logic while keeping state?

A. Bigger gas limit
B. More require checks
C. A longer timelock
D. Proxy delegation

Correct answer: D
A proxy that delegates calls to a separate logic contract lets the team swap the logic address while the proxy keeps the storage and address. A larger gas limit, extra guards, or a timelock do not make fixed logic replaceable after deployment.

Question 10 Domain 6: Security, Monitoring and Deployment

A validator runs on one server with no backup, and when it crashes block production stalls for the whole network. What best prevents this?

A. Run standby nodes
B. Raise the gas price
C. Delete old blocks
D. Lower the block time

Correct answer: A
Running redundant validator and standby nodes removes the single point of failure, so production continues when one server crashes. Gas price, pruning old blocks, and shorter block times do nothing to keep consensus alive during an outage.

CQRD-001. https://certlabz.com/certifications/certified-quorum-developer.html