CERTLABZ CERTIFICATION BOARD

Certified Ethereum Developer (CETD)

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 Ethereum 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: Ethereum Foundations — DATA

Sarah Miller maps each Ethereum concept to its role, with one role missing.

Concept	Role
Block	Ordered batch of transactions
State	(to choose)
Node	Validates and relays data
Consensus	Agrees the canonical chain

Table 1.1 Ethereum concepts and roles.

What does the state hold?

A. Only pending logs
B. Peer network map
C. Account balances and storage
D. Miner reward table

Correct answer: C
The Ethereum state records every account balance and the storage of each contract, updated as transactions execute. Blocks order transactions, nodes relay data, and consensus agrees the canonical chain, so none of those describe the state.

Question 2 Domain 2: Accounts, Transactions and Gas CONFIG

James Carter reviews a transaction object before it is signed and sent.

```
to:        0x9f3a...b21c
value:     1000000000000000000
gasLimit:  21000
maxFeePerGas: 30 gwei
nonce:     7
```

Listing 2.1 A transaction object.

What does the nonce prevent?

A High gas prices

B Replaying the transaction

C Contract reverts

D Slow block times

Correct answer: B
The nonce is the per-account transaction count, so each transaction is unique and an old signed one cannot be replayed. It does not set gas prices, stop reverts, or influence how fast blocks are produced.

Question 3 Domain 3: Solidity and Smart Contracts DATA

Emma Wilson maps each Solidity function visibility to its access, with one access missing.

Visibility	Access
public	Inside and outside
external	Only from outside
internal	(to choose)
private	This contract only

Table 3.1 Function visibility and access.

What access does internal give?

A Anyone off chain

B This and derived contracts

C Only the owner

D No caller at all

Correct answer: B
An internal function can be called from within this contract and from contracts that inherit from it, but not from outside. Public allows inside and outside, external only from outside, and private only within the one contract.

Question 4 Domain 3: Solidity and Smart Contracts CONFIG

David Brown reviews a snippet that tracks token balances by address.

```
mapping(address => uint) balances;

function deposit() public payable {
  balances[msg.sender] += msg.value;
}
```

Listing 4.1 A deposit function.

What does msg.sender refer to?

A. The contract address
B. The block miner
C. The zero address
D. The calling account

Correct answer: D
The value msg.sender is the address of the account or contract that called the current function, so the deposit is credited to the caller. It is not the contract itself, the miner, or the zero address.

Question 5 Domain 4: EVM and Tooling DATA

Olivia Davis compares the gas cost of EVM operations, with one entry missing.

Operation	Relative gas
ADD	Very low
SLOAD storage read	Moderate
SSTORE storage write	(to choose)
Memory access	Low

Table 5.1 Relative gas by operation.

What is the cost of a storage write?

A. High
B. Free
C. Same as ADD
D. Refunded always

Correct answer: A
Writing to contract storage with SSTORE is one of the most expensive EVM operations, so its relative gas cost is high. Arithmetic and memory access are cheap, and writes are never free or unconditionally refunded.

Question 6

Domain 4: EVM and Tooling

CONFIG

Ava Thompson lists the sections of a project build and test configuration.

```
solidity:  0.8.24
networks:  local, testnet
paths:     contracts, test
plugins:   ethers, coverage
```

Listing 6.1 A project configuration.

Which section sets the compiler version?

A networks
B paths
C plugins
D solidity

Correct answer: D
The solidity section pins the compiler version used to build the contracts, here 0.8.24. Networks lists chains to connect to, paths points to source folders, and plugins extends the tooling.

Question 7

Domain 5: dApp Development and Web3

A front end needs to read a contract balance and also display live updates when new deposits happen, without polling on a timer. What should the dApp use?

A A fixed page reload
B Manual refresh only
C Contract event listeners
D A slower poll loop

Correct answer: C
A dApp subscribes to contract events so the interface updates as soon as a matching log is emitted, with no timer polling. Manual refreshes, reloads, or slower polling all miss updates or waste requests.

Question 8

Domain 5: dApp Development and Web3

A user connects a browser wallet to a dApp and expects to approve each transaction before it is sent. What signs the transaction with the private key?

A. The dApp server
B. The wallet
C. The RPC node
D. The block proposer

Correct answer: B
The wallet holds the private key and signs the transaction after the user approves it, keeping the key off the dApp. The RPC node only relays the signed transaction, and the server and proposer never see the key.

Question 9 — Domain 6: Security, Testing and Deployment

A withdraw function sends ether to the caller before it updates the stored balance, letting a malicious contract call back in and drain funds. What flaw is this?

A. Integer overflow
B. Reentrancy
C. Gas underpricing
D. Stack too deep

Correct answer: B
Sending funds before updating state lets an attacker re-enter the function and withdraw again, which is a reentrancy attack. Updating state first, or using a guard, prevents it, unlike the unrelated overflow, gas, or stack issues.

Question 10 — Domain 6: Security, Testing and Deployment

A team is about to deploy a contract to mainnet and wants to catch logic bugs cheaply before real funds are at risk. What is the sound first step?

A. Deploy and watch
B. Skip all testing
C. Test on a testnet
D. Raise the gas limit

Correct answer: C
Deploying to a public testnet lets the team exercise the contract with no real value at stake, catching bugs before mainnet. Deploying straight to mainnet, skipping tests, or raising gas does nothing to surface logic flaws safely.

Published by the CertLabz Certification Board, a division of CertLabz LLC. Companion to objectives document CETD-001. https://certlabz.com/certifications/certified-ethereum-developer.html