CERTLABZ CERTIFICATION BOARD

Certified Solidity Developer (CSOD)

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 Solidity 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: Solidity Language Foundations

CONFIG

Sarah Miller reviews the opening of a contract before compiling it.

```
pragma solidity ^0.8.20;
contract Vault {
  uint256 total;
  function get() external view
    returns (uint256) { return total; }
}
```

Listing 1.1 A minimal contract.

Why can total be read but not written from outside?

A. It is a constant

B. It has no setter

C. It is payable

D. It is a mapping

Correct answer: B

The state variable total has default internal visibility and the only external function is a view getter, so no external code can write to it. It is not constant, payable, or a mapping, and it stays writable from inside the contract.

Question 2 — Domain 2: Types, Storage and Memory — DATA

James Carter maps each data location to how it behaves, with one row blank.

Location	Behavior
storage	Persists on chain
memory	(to choose)
calldata	Read only argument data
stack	Holds small local values

Table 2.1 Data locations and behavior.

What best describes memory?

A. Permanent contract state
B. Immutable input buffer
C. Temporary during a call
D. A signed event log

Correct answer: C
Memory is a temporary area that exists only for the duration of a single external call and is wiped afterward. Storage persists on chain, calldata is the read only argument buffer, and the stack holds small local values.

Question 3 — Domain 3: Functions, Modifiers and Events — CONFIG

Emma Wilson reviews a modifier applied to a restricted function.

```
modifier onlyOwner() {
  require(msg.sender == owner, "no");
  _;
}
function withdraw() external onlyOwner {
  payable(owner).transfer(address(this).balance);
}
```

Listing 3.1 A guarded function.

What does the underscore in the modifier mean?

A. A private variable
B. The body runs here
C. Skip the check
D. An unused label

Correct answer: B
The underscore marks where the body of the guarded function is inserted, so the require check runs first and the function body runs at that point. It is not a variable, a label, or a way to skip the check.

Question 4 — Domain 4: Contracts, Inheritance and Interfaces — DATA

Olivia Davis maps each construct to its role, with one description missing.

Construct	Role
interface	Declares functions only
abstract	Has unimplemented parts
library	(to choose)
contract	Full deployable unit

Table 4.1 Constructs and their roles.

What is the role of a library?

A. Holds contract state
B. Reusable stateless code
C. Signs transactions
D. Defines an interface

Correct answer: B
A library holds reusable stateless code that other contracts call, and it cannot hold its own state or receive ether. An interface declares functions, an abstract contract has unimplemented parts, and a contract is a full deployable unit.

Question 5 — Domain 5: Security and Common Vulnerabilities — CONFIG

David Brown reviews a withdraw function flagged during an audit.

```
function withdraw() external {
  uint256 bal = balances[msg.sender];
  (bool ok,) = msg.sender.call{value: bal}("");
  require(ok);
  balances[msg.sender] = 0;
}
```

Listing 5.1 A flagged withdraw.

Which flaw does this code contain?

A. Integer overflow
B. Missing constructor

C Reentrancy risk

D Bad visibility

Correct answer: C
The external call sends ether before the balance is zeroed, so an attacker contract can call back in and drain funds, which is reentrancy. Setting the balance to zero before the call, following checks effects interactions, closes the hole.

Question 6

Domain 6: Testing, Gas Optimization and Deployment

DATA

Ava Thompson compares two versions of a loop by gas per call, one value blank.

Version	Gas per call
Reads length each pass	(to choose)
Caches length once	Lower
Uses storage in loop	Highest
Uses memory in loop	Lower

Table 6.1 Loop patterns by gas cost.

What is the gas cost of rereading length each pass?

A Higher

B Free

C Refunded

D Always least

Correct answer: A
Rereading a stored length on every pass repeats a costly load, so it burns more gas than caching the length in a local variable once. Caching the length and working in memory both lower the gas the loop uses.

Question 7

Domain 1: Solidity Language Foundations

A developer wants a value that is set once at deploy time and can never change, while saving gas over a normal state variable read. Which choice fits best?

A A public mapping

B An immutable variable

C A memory variable

D A payable address

Correct answer: B
An immutable variable is assigned once in the constructor and then fixed, and it is cheaper to read than a normal storage slot. A mapping and a memory variable do not lock a single deploy time value, and payable is unrelated.

Question 8 Domain 3: Functions, Modifiers and Events

A contract needs off chain services to detect when a deposit happens, without those services scanning every storage slot. What is the right tool?

A. Emit an event
B. Return a value
C. Write to storage
D. Revert the call

Correct answer: A
Emitting an event writes an indexed log that off chain services can subscribe to cheaply, which is the standard way to signal a deposit. A return value is not visible off chain after mining, and storage writes are costly to scan.

Question 9 Domain 4: Contracts, Inheritance and Interfaces

A team wants one contract to call another whose source it does not control, knowing only the function signatures it must invoke. What should it use?

A. A full copy of it
B. An interface
C. A new library
D. An abstract base

Correct answer: B
An interface declares just the external function signatures, which lets one contract call another by address without holding its full source. Copying the source, a library, or an abstract base do not fit calling an external unknown contract.

Question 10 Domain 5: Security and Common Vulnerabilities

An auditor finds that any address can call a function that mints new tokens, because the function has no caller check at all. What is the core problem?

A. Integer underflow
B. Gas griefing
C. Broken access control
D. A stale oracle

Correct answer: C
The mint function lacks any restriction on who may call it, so this is broken access control that lets anyone create tokens. Adding an owner or role check gates the function, while overflow, oracle staleness, and gas griefing are different issues.

CSOD-001. https://certlabz.com/certifications/certified-solidity-developer.html