CERTLABZ CERTIFICATION BOARD

Certified PHP Programmer (CPHP)

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 PHP programmer 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: PHP Fundamentals and Syntax — CONFIG

Sarah Miller runs this short script and checks what it prints.

```
$name = "Sarah";
echo "Hi, $name";
```

Listing 1.1 A short echo script.

What does the script output?

A. Hi, $name
B. Hi, Sarah
C. Just the word Hi
D. A parse error

Correct answer: B
Inside double quotes PHP replaces a variable with its value, so $name becomes Sarah and the output is Hi, Sarah. Single quotes would print the literal text, and the script has no syntax error.

Question 2 — Domain 2: Data Types, Strings and Arrays — DATA

James Carter maps each array function to what it returns, with one entry missing.

Function	Returns
count($a)	Number of items
array_keys($a)	(to choose)
array_values($a)	The values only
in_array($v, $a)	True or false

Table 2.1 Array functions and results.

What does array_keys return?

A. The item count
B. The last value
C. The array keys
D. A sorted copy

Correct answer: C
The array_keys function returns all the keys of an array as a new indexed array. The count function returns the number of items, array_values returns the values, and in_array reports whether a value is present.

Question 3

Domain 3: Control Flow and Functions

CONFIG

Emma Wilson traces this loop by hand before running it.

```
$sum = 0;
for ($i = 1; $i <= 3; $i++) {
  $sum += $i;
}
echo $sum;
```

Listing 3.1 A counting loop.

What value is printed?

A. 3
B. 5
C. 6
D. 7

Correct answer: C
The loop adds 1, then 2, then 3 to $sum, which totals 6 when the counter passes 3 and the loop ends. It runs three times because the condition holds while $i is 1, 2 and 3.

Question 4

Domain 4: Object-Oriented Programming

CONFIG

Olivia Davis reviews two classes where the child redefines a method.

```
class Animal {
  public function speak() { return "..."; }
}
class Dog extends Animal {
  public function speak() { return "Woof"; }
}
echo (new Dog)->speak();
```

Listing 4.1 A class and a subclass.

What does the last line print?

A. Three dots
B. Woof
C. An empty string
D. A fatal error

Correct answer: B
Dog overrides the speak method, so calling it on a Dog object runs the child version and prints Woof. The parent method is replaced for Dog instances, and both methods are public so no error occurs.

Question 5

Domain 5: Web Requests, Sessions and the Standard Library

DATA

Liam Anderson maps each superglobal to the data it holds, with one entry missing.

Superglobal	Holds
$_GET	URL query data
$_POST	(to choose)
$_SESSION	Session values
$_COOKIE	Cookie values

Table 5.1 Superglobals and their data.

What does $_POST hold?

A. URL query data
B. Server settings
C. The session id
D. Posted form data

Correct answer: D
The $_POST array holds values sent in the body of a form submitted with the post method. Query string values arrive in $_GET, session values in $_SESSION, and cookie values in $_COOKIE.

Question 6 — Domain 6: Errors, Security and Best Practices — CONFIG

Ava Thompson stores a hash at sign-up and checks it at login.

```
$hash = password_hash($pw, PASSWORD_DEFAULT);
// later, at login
if (password_verify($input, $hash)) {
  // grant access
}
```

Listing 6.1 Hashing and checking a password.

What does password_verify do here?

A. Checks a password
B. Hashes a password
C. Encrypts a file
D. Starts a session

Correct answer: A
The password_verify function checks whether the entered password matches the stored hash and returns true when it does. It does not create a hash itself, which is the job of password_hash performed earlier at sign-up.

Question 7 — Domain 1: PHP Fundamentals and Syntax

A script uses require to load a config file that is missing, and the whole script halts with a fatal error rather than carrying on. What does this show about require?

A. It warns and goes on
B. It stops on failure
C. It ignores the file
D. It caches the file

Correct answer: B
The require statement treats a missing file as a fatal error and stops the script, because the file is considered essential. The include statement only raises a warning and lets the script continue, which is the key difference.

Question 8 — Domain 3: Control Flow and Functions

A variable set inside a function cannot be read by code after the call, and it disappears when the function returns. What best explains this behaviour?

A. Global by default
B. Static binding
C. Type juggling

D Local scope

Correct answer: D
Variables created inside a function have local scope, so they exist only during the call and are not visible outside it. Reaching outer values needs the global keyword or a parameter, and neither type juggling nor binding is involved.

Question 9

Domain 5: Web Requests, Sessions and the Standard Library

A site must remember a signed-in user across many page requests without asking for the password on every page. Which feature fits this need best?

A Sessions
B A new constant
C A global var
D A local var

Correct answer: A
Sessions keep per-user data on the server across requests, tied to the visitor by a session id, so a signed-in state persists between pages. A constant or a variable lives only for a single request and cannot carry state forward.

Question 10

Domain 6: Errors, Security and Best Practices

User input is placed straight into an SQL string, which lets an attacker change the meaning of the query. What best prevents this kind of attack?

A Longer passwords
B Bound parameters
C More code comments
D A faster server

Correct answer: B
Prepared statements with bound parameters keep user input separate from the SQL code, so input can never alter the query structure. Longer passwords, extra comments, or a faster server do nothing to stop SQL injection.

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