CERTLABZ CERTIFICATION BOARD

# Certified Swift Programmer (CSWP)

## 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 Swift 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: Swift Fundamentals and Syntax

CONFIG

Sarah Miller reviews a short Swift snippet that declares two values.

```
let name = "Sarah"
var age = 30
age = 31
name = "Emma"
```

Listing 1.1 Four lines of Swift.

Which line fails to compile?

A. age = 31

B. name = "Emma"

C. var age = 30

D. let name = "Sarah"

**Correct answer: B**

A value declared with let is a constant, so reassigning name after it is set will not compile. The var binding age can change freely, which is why the age line and the two declarations are all valid.

## Question 2 — Domain 2: Types, Optionals and Collections — CONFIG

James Carter checks the inferred type of a converted value.

```
let text = "42"
let number = Int(text)
```

Listing 2.1 Converting a string.

What is the type of number?

A. Int
B. String
C. Int?
D. Double

**Correct answer: C**
The Int(text) initialiser can fail when the string is not a valid number, so it returns an optional and number is Int?. A plain Int would be wrong because the conversion is not guaranteed to succeed.

## Question 3 — Domain 3: Control Flow and Functions — CONFIG

Olivia Davis reads a function that uses a default parameter.

```
func greet(_ name: String,
           with word: String = "Hi")
  -> String {
  return word + ", " + name
}
let msg = greet("David")
```

Listing 3.1 A default parameter.

What is the value of msg?

A. David
B. word, David
C. Hi, David
D. nil

**Correct answer: C**
The call omits the with argument, so Swift uses the default value Hi and returns Hi, David. The word placeholder is a parameter name, not literal text, and the function always returns a non-optional String.

## Question 4 — Domain 4: Structs, Classes and Protocols — DATA

Liam Anderson compares struct and class behaviour, with one cell missing.

| Feature | struct | class |
|---|---|---|
| Category | Value type | Reference type |
| Inheritance | No | (to choose) |
| Copy on assign | Yes | No |
| Identity check | No | Yes |

Table 4.1 Struct against class.

What fills the missing cell?

A. Yes
B. Never
C. Value only
D. Only via protocols

**Correct answer: A**
A class can inherit from another class, so the missing cell is Yes. Structs cannot inherit from other structs, which is why the struct column shows No, and inheritance is not limited to protocols.

## Question 5

Domain 5: Closures, Generics and the Standard Library

CONFIG

Ava Thompson traces a chained call on an array of numbers.

```
let nums = [1, 2, 3, 4]
let result = nums
  .filter { $0 % 2 == 0 }
  .map { $0 * 10 }
```

Listing 5.1 Filter then map.

What is result?

A. [10, 20, 30, 40]
B. [2, 4]
C. [10, 30]
D. [20, 40]

**Correct answer: D**
The filter keeps only the even numbers 2 and 4, then map multiplies each by ten to give [20, 40]. The odd numbers are dropped before the map runs, so results built from 1 and 3 are wrong.

## Question 6

Domain 6: Error Handling, Memory and Best Practices

DATA

David Brown maps each error handling keyword to its meaning, with one missing.

| Keyword | Meaning |
| --- | --- |
| throws | Function may throw |
| try | (to choose) |
| catch | Handle the error |
| defer | Run on scope exit |

Table 6.1 Error handling keywords.

What does try mean?

A. Retry on failure
B. Call throwing code
C. Suppress the error
D. Define a new error type

**Correct answer: B**
The try keyword marks a call to code that can throw, so control can jump to a catch when an error is raised. It does not retry the call, hide the error, or declare a new error type on its own.

## Question 7 Domain 1: Swift Fundamentals and Syntax

A developer writes let pi = 3.14 with no type annotation, and the compiler treats pi as a Double. What Swift feature assigns the type here?

A. Type inference
B. Type casting
C. Dynamic dispatch
D. Runtime type checking

**Correct answer: A**
Type inference lets the compiler pick the type from the literal, so pi becomes a Double without an annotation. Casting converts between known types, and dispatch and runtime checks concern method calls, not this assignment.

## Question 8 Domain 3: Control Flow and Functions

Inside a function a developer wants to exit early when an optional input is nil, while keeping the unwrapped value in scope for the rest of the function. Which statement fits best?

A. if let block
B. switch statement
C. guard let else
D. nested if let blocks

**Correct answer: C**
A guard let with an else exits the function early when the value is nil and keeps the unwrapped value available afterward. An if let confines the value to its own block, and a switch does not fit a simple early exit.

## Question 9

Domain 4: Structs, Classes and Protocols

A team wants several unrelated types to share a common set of method requirements without forcing them into one class hierarchy. What Swift construct fits best?

A. A protocol
B. A base class
C. A global function
D. A stored property

**Correct answer: A**
A protocol defines a set of requirements that many unrelated types can adopt without sharing a base class. A base class would force a hierarchy, and a function or property does not describe shared requirements across types.

## Question 10

Domain 6: Error Handling, Memory and Best Practices

Two class instances hold strong references to each other, so neither is ever released and the memory leaks. What best breaks the cycle?

A. Add more classes
B. A weak reference
C. Copy both objects
D. Disable ARC fully

**Correct answer: B**
Marking one side of the pair as weak stops it from keeping the other alive, so ARC can release both when they are no longer used. Adding classes or copying objects does not remove the cycle, and ARC cannot be turned off.

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