University of Toronto, Department of Electrical and Computer Engineering

ECE 1724-F1 — Performant Software Systems with Rust

Quiz 4 — Sample Quiz 2 — Answer Key

Wednesday, November 11, 2026

1. Ⓑ **A method can introduce its own type parameter.**

The method's `U` is independent of the enclosing `W`. Here `W` is `u32`, but `U` is `bool`, so the returned wrapper contains `false`. (Book 10.1.) Teaching variation of `CappedWriter<W>`.

2. Ⓓ **A default method can call a required method.**

`Probe` supplies `code`, which returns `6`. The inherited `report` method calls that implementation and adds one, yielding `7`. (Book 10.2.) Teaching variation of `CommandRunner`.

3. Ⓐ **One type parameter denotes one concrete type per call.**

Implementing the same trait does not make `Live` and `Dry` the same type. The repeated `T` requires both arguments to agree. Separate type parameters, or two independent `impl EditorEffects` parameters, would permit this call. (Book 10.2.) Teaching variation of the `EditorEffects` bound.

4. Ⓒ **The associated type fixes the method signature.**

Here `Self::Item` is `char`, so `next` must return `Option<char>`. The declared return type `Option<u32>` does not match. To keep the chosen item type, use `Option<char>` and return a character, such as `Some('A')`. (Book 20.2, Associated Types.) Custom teaching trait inspired by `Iterator<Item = char>`.

5. Ⓑ **A child-trait bound includes its supertrait requirements.**

Every `WorkspaceEffects` implementor must implement `EditorEffects`. Therefore `total` may call `count` without repeating the parent bound. `Log` returns `4`, giving `6`. Both trait implementations are explicitly supplied. (Book 20.2, Supertraits.) Teaching variation of `WorkspaceEffects`.

Book sections: *The Rust Programming Language*, 10.1: Generic Data Types, 10.2: Traits, and 20.2: Associated Types and Supertraits only.