**University of Toronto, Department of Electrical and Computer Engineering**
**ECE 1724-F1 — Performant Software Systems with Rust**

# Quiz 4 — Sample Quiz 1 — Answer Key

Wednesday, November 11, 2026

**1.** Ⓒ **Different instances can use different type arguments.**

Here `a` has type `CappedWriter<u32>` and `b` has type `CappedWriter<bool>`. The generic method returns the corresponding inner type for each instance. (Book 10.1.)

**2.** Ⓓ **An implementation may keep or replace a default.**

`Noop` keeps the trait's default method, returning `0`. `Recorder` replaces that method with its own body, returning `3`. An empty implementation is sufficient when all trait methods have defaults. (Book 10.2.)

**3.** Ⓑ **Generic bodies need declared trait bounds.**

The body must type-check using the declared capabilities of `E`. This particular caller does not supply a missing bound. Writing `fn count_events<E: EditorEffects>` permits the method call and produces `4`. (Book 10.2.)

**4.** Ⓐ **An associated-type constraint fixes the item type.**

`Iterator<Item = char>` makes `next()` return `Option<char>`. `Item` is an associated type, not a generic parameter to `Iterator`. The unconstrained bound does not guarantee the required return type. (Book 20.2, Associated Types.)

**5.** Ⓒ **Every supertrait requirement must be satisfied.**

Implementing `WorkspaceEffects` requires both `EditorEffects` and `Clone`. Only the former is implemented. Adding `#[derive(Clone)]` to `Recorder` supplies the missing requirement; the default method can stay unchanged. (Book 20.2, Supertraits.)

**Sources and adaptations:** Examples are simplified teaching variations of Rustrace at commit `6b57e41002ff`. Q1 retains its generic wrapper pattern; Q2 changes the hook signature to expose default and overridden return values; Q3 intentionally removes a generic trait bound; Q4 retains an iterator-item constraint; Q5 retains the supertrait requirement while omitting `Clone`. Q1: CappedWriter, Q2: EditorEffects, Q3: replay_transactions, Q4: utf16_at_byte, Q5: WorkspaceEffects.

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