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

Quiz 4 — Instructor Answer Key

Wednesday, November 11, 2026

1. Ⓓ **Declare the generic parameter on the implementation.**

`impl<W> CappedWriter<W>` introduces `W` and uses it as the struct's argument. A and B leave the method's `W` undeclared; C omits the struct's required type argument. The completed program prints `6`. (Book 10.1.)

2. Ⓐ **Only methods with bodies have defaults.**

`enabled` has a default body, but `record` has only a signature. The implementation must provide `record`, even though this program never calls it. No `Clone` bound is present. (Book 10.2.)

3. Ⓒ **Separate type parameters may have different types.**

`A` is `System` and `B` is `Fake`. Both implement `CommandRunner`, so both calls are valid and return `2` and `5`. Their sum is `7`. (Book 10.1–10.2.)

4. Ⓑ **The implementation fixes the associated item type.**

`type Item = char` makes `Option<Self::Item>` mean `Option<char>` in this implementation. Debug formatting prints `Some('r')`; no type argument is needed on `next`. (Book 20.2, Associated Types.)

5. Ⓓ **Every supertrait obligation must be satisfied.**

The derive supplies `Clone`, but `Recorder` still needs a separate `impl EditorEffects for Recorder {}`. Implementing the child trait does not implement its parents automatically. With that addition, the default method would print `false`. (Book 20.2, Supertraits.)

Book scope: *The Rust Programming Language*, 10.1, 10.2, and 20.2 (Associated Types and Supertraits only).