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

Quiz 2 — Answer Key to Sample Questions

Wednesday, September 30, 2026

1. Ⓒ **Cloning and returning ownership.**

The clone is independent of `original`. It moves into `label`, is modified, then moves back into `tagged`. The original remains `ready`. (Book 4.1.)

2. Ⓐ **A borrow ends after its last use.**

The immutable borrow through `view` ends after the first print, so the mutable borrow is allowed. `edit` already has type `&mut String`; its binding need not be mutable to modify the referenced string. (Book 4.2.)

3. Ⓓ **Overlapping mutable borrows.**

`first` is used after `second` is created, so both mutable borrows would overlap. Rust rejects this at compile time; nothing is printed. (Book 4.2.)

4. Ⓑ **A slice borrows the original data.**

`word` is an immutable slice into `text`, not a copy. Its use after `clear()` keeps that borrow active across a conflicting mutable borrow. (Book 4.3.)

5. Ⓐ **Appending a slice and a character.**

`push_str` borrows `ending` and appends its contents; `push` appends one character. Only `text` changes, producing `rainbow! bow`. (Book 8.2.)

6. Ⓒ **Concatenation and formatting.**

`+` consumes its left `String` and borrows its right operand. `format!` borrows the strings here and creates `caption`, containing `rise: sunrise`. `right` and `joined` remain usable. (Book 8.2.)

7. Ⓓ **Byte length and UTF-8 boundaries.**

The accented character uses two bytes and `lan` uses three, so `len()` returns `5`. The range ends inside the first character, causing a run-time panic at the slice expression, before the final print. (Book 4.3 and 8.2.)

8. Ⓑ **Field initialization and mutability.**

Field order need not match the definition. The shorthand `name` means `name: name`, moving the string into the struct. The mutable binding allows the field assignment; individual fields are not declared `mut`. (Book 5.1.)

9. Ⓒ **Struct update can move individual fields.**

The update moves `first.name` into `second.name`. The unmoved `bool` field `first.active` remains accessible, so C prints `true Mina`. A uses the moved field; B uses the partially moved whole value; D tries to move the name again. (Book 5.1 and 4.1.)

10. Ⓐ **Tuple structs define distinct types.**

The function requires a `Position`, but `speed` is a `Velocity`. Identical field types do not make these struct types interchangeable. Calling `show(Position(speed.0, speed.1))` would work. (Book 5.1.)

Sources: *The Rust Programming Language*, Chapter 4, Section 8.2, and Section 5.1.