University of Toronto, Department of Electrical and Computer Engineering

ECE 1724-F1 — Performant Software Systems with Rust

Quiz 4 — Sample Quiz 2

Wednesday, November 11, 2026

Note: Each question has exactly one correct answer. All snippets are complete programs; ignore compiler warnings. The examples are simplified teaching variations inspired by Rustrace; no repository knowledge is required.

1. What does this complete program print?

```
struct CappedWriter<W> { inner: W }
impl<W> CappedWriter<W> {
    fn replace<U>(self, inner: U) -> CappedWriter<U> {
        CappedWriter { inner }
    }
}
fn main() {
    let writer = CappedWriter { inner: 7u32 };
    let changed = writer.replace(false);
    println!("{}", changed.inner);
}
```

Ⓐ `7`

Ⓑ `false`

Ⓒ `true`

Ⓓ It does not compile because `U` must equal `W`.

2. What does this complete program print?

```
trait CommandRunner {
    fn code(&self) -> u32;
    fn report(&self) -> u32 { self.code() + 1 }
}
struct Probe;
impl CommandRunner for Probe {
    fn code(&self) -> u32 { 6 }
}
fn main() {
    println!("{}", Probe.report());
}
```

Ⓐ `6`

Ⓑ `0`

Ⓒ It does not compile because a default method cannot call a required method.

Ⓓ `7`

3. Why does this complete program fail to compile?

```
trait EditorEffects {}
struct Live;
struct Dry;
impl EditorEffects for Live {}
impl EditorEffects for Dry {}
fn pair<T: EditorEffects>(first: T, second: T) {}
fn main() {
    pair(Live, Dry);
}
```

(A) Both parameters use the same `T`, so they require the same concrete type.

(B) An empty trait cannot be implemented.

(C) Both structs must derive `Clone`.

(D) A generic function cannot have two parameters.

4. Why does this complete program fail to compile?

```
trait EventSource {
    type Item;
    fn next(&mut self) -> Option<Self::Item>;
}
struct Characters;
impl EventSource for Characters {
    type Item = char;
    fn next(&mut self) -> Option<u32> { Some(65) }
}
fn main() {}
```

(A) An associated type cannot be a primitive type.

(B) Every associated type must be named `Output`.

(C) The trait requires `Option<char>`, but the method returns `Option<u32>`.

(D) The trait declaration must provide a default method body.

5. What does this complete program print?

```
trait EditorEffects { fn count(&self) -> u32; }
trait WorkspaceEffects: EditorEffects {}
struct Log;
impl EditorEffects for Log {
    fn count(&self) -> u32 { 4 }
}
impl WorkspaceEffects for Log {}
fn total<T: WorkspaceEffects>(effects: &T) -> u32 {
    effects.count() + 2
}
fn main() {
    println!("{}", total(&Log));
}
```

Ⓐ `4`

Ⓑ `6`

Ⓒ It does not compile because `total` needs an additional explicit `EditorEffects` bound.

Ⓓ It does not compile because a trait cannot require another trait.