University of Toronto, Department of Electrical and Computer Engineering

ECE 1724-F1 — Performant Software Systems with Rust

Quiz 4 — Sample Quiz 1

Wednesday, November 11, 2026

Note: Each question has exactly one correct answer. Unless a complete program is shown, assume the code is inside `fn main()`. Ignore compiler warnings. These self-contained teaching adaptations of Rustrace require no knowledge of its codebase.

1. What does this complete program print?

```
struct CappedWriter<W> { inner: W }

impl<W> CappedWriter<W> {
    fn into_inner(self) -> W { self.inner }
}

fn main() {
    let a = CappedWriter { inner: 8u32 };
    let b = CappedWriter { inner: true };
    println!("{} {}", a.into_inner(), b.into_inner());
}
```

Ⓐ `8 1`

Ⓑ It does not compile because both instances must use the same `W`.

Ⓒ `8 true`

Ⓓ It does not compile because a generic method cannot return `W`.

2. What does this complete program print?

```
trait EditorEffects {
    fn record_replay(&self) -> u32 { 0 }
}
struct Noop;
impl EditorEffects for Noop {}
struct Recorder;
impl EditorEffects for Recorder {
    fn record_replay(&self) -> u32 { 3 }
}
fn main() {
    println!("{} {}", Noop.record_replay(),
                      Recorder.record_replay());
}
```

Ⓐ `0 0`

Ⓑ It does not compile because `Noop` has an empty trait implementation.

Ⓒ `3 3`

Ⓓ `0 3`

3. Why does this complete program fail to compile?

```
trait EditorEffects {
    fn count(&self) -> u32;
}
struct Recorder;
impl EditorEffects for Recorder {
    fn count(&self) -> u32 { 4 }
}
fn count_events<E>(effects: &E) -> u32 {
    effects.count()
}
fn main() {
    println!("{}", count_events(&Recorder));
}
```

Ⓐ Trait methods cannot return integers.

Ⓑ The generic parameter `E` needs an `EditorEffects` bound.

Ⓒ `Recorder` must be a mutable binding.

Ⓓ Generic functions cannot accept references.

4. Which bound makes this complete program compile? Here `"rust".chars()` produces an iterator of `char` values.

```
fn first<I: /* bound */>(mut items: I) -> Option<char> {
    items.next()
}
fn main() {
    println!("{:?}", first("rust".chars()));
}
```

Ⓐ `Iterator<Item = char>`

Ⓑ `Iterator<char>`

Ⓒ `Iterator<Item = u8>`

Ⓓ `Iterator`

5. Why does this complete program fail to compile?

```
trait EditorEffects {}
trait WorkspaceEffects: EditorEffects + Clone {
    fn command_active(&self) -> bool { false }
}
struct Recorder;
impl EditorEffects for Recorder {}
impl WorkspaceEffects for Recorder {}

fn main() {}
```

(A) A trait cannot have more than one supertrait.

(B) `command_active` must be reimplemented even though it has a default.

(C) `Recorder` does not implement the required supertrait `Clone`.

(D) `Recorder` needs a field before it can implement any trait.