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

Quiz 4 — Multiple Choice Questions

Wednesday, November 11, 2026

Name: _____

Student Number: _____

Instructions: Choose exactly one answer per question. Each program is independent. Ignore compiler warnings. The examples are simplified from Rustrace; no knowledge of the project is required.

1. Which replacement for `/* header */` makes this program compile?

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

/* header */ {
    fn into_inner(self) -> W { self.inner }
}

fn main() {
    let writer = CappedWriter { inner: 6_u32 };
    println!("{}", writer.into_inner());
}
```

Ⓐ `impl CappedWriter<W>`
Ⓑ `impl CappedWriter<u32>`
Ⓒ `impl<W> CappedWriter`
Ⓓ `impl<W> CappedWriter<W>`

2. What happens when this program is compiled and run?

```
trait EditorEffects {
    fn record(&mut self);
    fn enabled(&self) -> bool { true }
}

struct Noop;

impl EditorEffects for Noop {}

fn main() {
    println!("{}", Noop.enabled());
}
```

Ⓐ It does not compile: the required `record` method is missing.
Ⓑ It prints `true`.
Ⓒ It does not compile: `Noop` must implement `Clone`.
Ⓓ It prints `false`.

3. What does this complete program print?

```
trait CommandRunner {
    fn status(&self) -> u32;
}

struct System;
struct Fake;

impl CommandRunner for System {
```

```
    fn status(&self) -> u32 { 2 }
}

impl CommandRunner for Fake {
    fn status(&self) -> u32 { 5 }
}

fn total<A: CommandRunner, B: CommandRunner>(a: A, b: B) -> u32 {
    a.status() + b.status()
}

fn main() {
    println!("{}", total(System, Fake));
}
```

(A) 2

(B) 5

(C) 7

(D) It does not compile: both arguments must have the same concrete type.

4. What does this complete program print?

```
struct Cursor;

impl Iterator for Cursor {
    type Item = char;
    fn next(&mut self) -> Option<Self::Item> { Some('r') }
}

fn main() {
    let mut cursor = Cursor;
    println!("{:?}", cursor.next());
}
```

(A) `r`

(B) `Some('r')`

(C) `Some(114)`

(D) It does not compile: `next` needs an explicit type argument.

5. Why does this complete program fail to compile?

```
trait EditorEffects {}

trait WorkspaceEffects: EditorEffects + Clone {
    fn command_active(&self) -> bool { false }
}

#[derive(Clone)]
struct Recorder;

impl WorkspaceEffects for Recorder {}

fn main() {
    println!("{}", Recorder.command_active());
}
```

(A) A trait cannot require two supertraits.

(B) `Clone` cannot be derived for a struct without fields.

(C) Every trait method must be overridden.

(D) `Recorder` does not implement `EditorEffects`.