University of Toronto, Department of Electrical and Computer Engineering

ECE 1724-F1 — Performant Software Systems with Rust

Quiz 5 — Sample Quiz 1

Wednesday, November 25, 2026

Instructions: Choose exactly one answer per question. Each complete program is independent. Ignore compiler warnings; assume Rust 2024. These teaching examples use standard-library APIs and patterns; no knowledge of library internals is required.

1. What does this complete program print? trim removes whitespace from both ends and returns a borrowed string slice.

```
fn clean(text: &str) -> &str {
    text.trim()
}
fn main() {
    let line = String::from("  ready  ");
    let word = clean(&line);
    println!("{} {}", word, line.len());
}
```

Ⓐ ready 5

Ⓑ It fails to compile because clean needs an explicit lifetime.

Ⓒ ready 9

Ⓓ It fails to compile because trim consumes line.

2. Why does this complete program fail to compile?

```
fn label<'a>() -> &'a str {
    let text = String::from("saved");
    text.as_str()
}
fn main() {
    println!("{}", label());
}
```

Ⓐ The returned slice refers to text, which is dropped when label returns.

Ⓑ String values cannot be created inside functions.

Ⓒ Every lifetime must be named static.

Ⓓ as_str transfers ownership of text to the caller.

3. What does this complete program print? unwrap_or_else calls its closure only when the Option is None.

```
fn main() {
    let fallback = String::from("idle");
    let configured: Option<usize> = None;
    let size = configured.unwrap_or_else(|| fallback.len());
    println!("{} {}", size, fallback);
}
```

Ⓐ 0 idle

Ⓑ 4 idle

Ⓒ It fails to compile because the closure moves fallback.

Ⓓ 4 4

4. What does this complete program print? Each use of then below separates successive output lines.

```
fn main() {
    let values = vec![2, 5];
    let mut doubled = values.iter().map(|value| {
        println!("visit {}", value);
        value * 2
    });
    println!("start");
    println!("{:?}", doubled.next());
}
```

Ⓐ visit 2 then visit 5 then start then Some(4)

Ⓑ start then visit 2 then visit 5 then Some(4)

Ⓒ start then Some(4)

Ⓓ start then visit 2 then Some(4)

5. What does this complete program print? Here, * accesses the value stored in a box.

```
fn main() {
    let mut stored = Box::new(7);
    let copied = *stored;
    *stored += 3;
    println!("{} {}", copied, *stored);
}
```

Ⓐ 10 10

Ⓑ 7 10

Ⓒ It fails to compile because a Box cannot be mutable.

Ⓓ It prints the address of stored.