University of Toronto, Department of Electrical and Computer Engineering

ECE 1724-F1 — Performant Software Systems with Rust

Quiz 5 — Sample Quiz 2

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?

```
fn shorter<'a>(left: &'a str, right: &'a str) -> &'a str {
    if left.len() < right.len() {
        left
    } else {
        right
    }
}
fn main() {
    let outer = String::from("station");
    {
        let inner = String::from("bus");
        let chosen = shorter(&outer, &inner);
        println!("{}", chosen);
    }
}
```

Ⓐ station

Ⓑ It fails to compile because the two Strings were created in different scopes.

Ⓒ bus

Ⓓ It requires a static lifetime.

2. What does this complete program print? trim returns a slice without changing the String.

```
struct Entry<'a> {
    text: &'a str,
}
fn main() {
    let line = String::from("  green ");
    let entry = Entry { text: line.trim() };
    println!("{} {}", entry.text, line.len());
}
```

Ⓐ green 5

Ⓑ It fails to compile because a struct cannot store a reference.

Ⓒ It fails to compile because Entry needs to own a String.

Ⓓ green 8

3. Why does this complete program fail to compile?

```
fn main() {
    let identity = |value| value;
    let name = identity(String::from("east"));
    let count = identity(4_u32);
    println!("{} {}", name, count);
}
```

(A) The first call fixes the closure parameter and return type to String.

(B) Closures must always have explicit parameter types.

(C) A closure can be called only once.

(D) Closures cannot return their arguments.

4. What does this complete program print?

```
fn main() {
    let limit = 3;
    let values = vec![2, 4, 6];
    let selected: Vec<i32> = values
        .into_iter()
        .filter(|value| *value > limit)
        .map(|value| value * 2)
        .collect();
    println!("{:?}", selected);
}
```

(A) [4, 8, 12]

(B) [8, 12]

(C) [4, 6]

(D) [12]

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

```
enum List {
    Item(i32, Box<List>),
    End,
}
fn main() {
    let list = List::Item(3, Box::new(List::Item(5, Box::new(List::End))));
    let total = match list {
        List::Item(first, rest) => match *rest {
            List::Item(second, _) => first + second,
            List::End => first,
        },
        List::End => 0,
    };
    println!("{}", total);
}
```

Ⓐ 3

Ⓑ It fails to compile because recursive types are never allowed.

Ⓒ 5

Ⓓ 8