Advanced Rust 2026 - Lab 3: Macro Rules in Practice

Lukáš Hozda

Spring 2026

Lab Goals

1. Build robust ‘macro$_{\text{rules}}$¡ patterns.
2. Practice diagnostics and fallback arms.
3. Prepare for homework pair A3.

Time Plan (90 min)

1. 15 min - recap (‘macro$_{\text{rules}}$¡ execution model)
2. 25 min - exercise 1 (repetition patterns)
3. 25 min - exercise 2 (command DSL macro)
4. 15 min - exercise 3 (generated tests)
5. 10 min - review

Exercise 1: Repetition Macro

Create macro ‘vec$_{\text{ofstrings}}$¡:

```
let v = vec_of_strings![10, 20, 30];
assert_eq!(v, vec!["10", "20", "30"]);
```

Requirements:

1. Accept comma-separated expressions.
2. Preserve order.
3. Trailing comma optional.

Exercise 2: Mini Command DSL

Create ‘calc¡ macro with supported forms:

1. ‘calc!(add a, b)‘
2. ‘calc!(sub a, b)‘
3. ‘calc!(mul a, b)‘

Invalid forms must trigger clear ‘compile$_{\text{error}}$¡.

Exercise 3: Test Case Generator

Write macro that generates repetitive tests from tuples:

```
case!(sum_small, [1,2,3], 6);
```

Debrief

1. Where is macro syntax better than function API?
2. Which diagnostics were easy for users to understand?
3. What should remain functions, not macros?