Student Name: ____________________________________ Student ID: ____________________ TA: _____________________________

Rust Lab 03

15/7/2025

1. Mastering Variables and Logic in Rust

You will write several small Rust programs to explore these topics.

Part 1: Variable Basics and Mutability

- Declare an immutable variable `year` with value 2025.
- Declare a mutable variable `month` with value 7, then reassign it to 12.
- Print `year` and `month` on the same line.
- Try reassigning `year` to another value and fix the resulting error.

TA Note: ____

Part 2: Type Inference and Consistency

- Declare variable `price = 99.99`.
- Try assigning `price = 100` and explain the compiler error.
- Fix the mismatch using appropriate Rust syntax.
- Create a boolean `discount_applied = price < 100.0` and print the result.

TA Note: ____

Part 3: Logic Expressions and Boolean Connectives

- Use variables: `available = true`, `in_stock = false`, `rating = 4.5`.
- Create `is_good = available && rating > 4.0 && in_stock`.
- Print `is_good`, modify `in_stock = true`, and test again.
- Print results of `!available`, `available || in_stock`, `rating < 3.0 || rating > 4.0`.

TA Note: ____

Part 4: Shadowing and Redeclaration

- Declare `let mut score = 80;`, reassign with `score + 10`.
- Shadow `score` as `let score = score > 85;`.
- Shadow again with `let score = if score { "Passed" } else { "Failed" };`
- Print the final value of `score`.

TA Note: ____

Part 5: Scope and Lifetime

- Declare `let a = 10;` in main block.
- Create inner block `{ let b = a + 5; println!(...) }`.
- Try printing `b` outside the block and fix the scope issue using shadowing.
- Demonstrate variable shadowing with type and mutability change in nested block.

TA Note: ____

Part 6: Code Review and Commenting

- Add comments explaining purpose of each variable.
- Explain logic and shadowing.
- Describe how type or scope errors were resolved.

Student Name: ________________________________ Student ID: __________________ TA: __________________________

TA Note: ____

2. Event Ticket Discount Checker

Develop a Rust program to simulate a discount checker for an event ticket booking system. This exercise reinforces concepts of Chapter 3 n book: variable mutability, type consistency, shadowing, boolean logic, and scope. Do not use constructs outside this chapter.

Requirements

2.1) Declare immutable variable `base_price` with value 150.0.

2.2) Declare mutable variable `discount` and calculate total discount:
- 10% if is_student is true
- 20% if is_early_bird is true
- 5% if has_coupon is true

2.3) Use boolean variables `is_student`, `is_early_bird`, `has_coupon`.

2.4) Use shadowing to calculate final_price after applying discount.

2.5) Use a block scope to check if final_price < 50.0 and set `free_entry` accordingly.

2.6) Print all variables and results as shown in the expected output.

2.7) Try printing `free_entry` outside its scope and comment the resulting compiler error.

Example Output

```
Base ticket price: $150.0
Student discount applied: true
Early bird discount applied: false
Coupon used: true
Final ticket price: $127.5
Free entry: false
```

Hints

• Use expressions like: if is_student { discount += 0.10; }

• Use only let, mut, shadowing, arithmetic, and print macros.

• Do not use control flow constructs like if/else for output branching.

TA Comment: ____

____