UNIVERSITY OF TORONTO
FACULTY OF APPLIED SCIENCE AND ENGINEERING

ECE 1724-F1 — Performant Software Systems with Rust

Sample Midterm 3 — Answer Key

Coverage: Rust Book Chapters 1–6, 8, and 9. Chapter 7 is excluded.

110 minutes; 100 marks. Section A: 20 questions × 2 marks. Section B: 20 questions × 3 marks.

Question 1: (D) 2 marks

Cargo.toml is the package manifest and records package metadata and declared dependencies. Cargo.lock records the exact dependency versions selected for the project; it is not where package metadata is normally specified. *Book: 1.3.*

Question 2: (C) 2 marks

rustc compiles the source file into an executable when compilation succeeds. Compilation and execution are separate steps; the resulting program must then be run to execute its main function. *Book: 1.2.*

Question 3: (D) 2 marks

read_line returns a Result indicating whether reading input succeeded. expect obtains the successful value or panics on an error. Parsing and trimming are separate operations performed on the input text. *Book: 2.*

Question 4: (B) 2 marks

continue skips the rest of the current iteration and starts the next iteration of the loop. This lets a guessing-game input loop request another guess after input cannot be parsed. *Book: 2; 3.5.*

Question 5: (A) 2 marks

Constants require an explicit type annotation and are always immutable. They cannot be combined with mut, and they may be declared in scopes outside main, including at module scope. *Book: 3.1.*

Question 6: (A) 2 marks

Arrays contain a fixed number of elements of a single type. Tuples also have a fixed length, but their individual fields may have different types, such as an integer and a boolean. *Book: 3.2.*

Question 7: (B) 2 marks

An if expression used to initialize a binding must have compatible branch types. The integer branch and string-slice branch do not match, even though this particular condition is true. The program cannot compile. *Book: 3.5.*

Question 8: (D) 2 marks

Cloning creates a separate owned String with its own contents. Appending to copy changes only that String; original remains go and is still available for the final print. *Book: 4.1.*

UNIVERSITY OF TORONTO
FACULTY OF APPLIED SCIENCE AND ENGINEERING

ECE 1724-F1 — Performant Software Systems with Rust

Sample Midterm 3 — Answer Key (continued)

Coverage: Rust Book Chapters 1–6, 8, and 9. Chapter 7 is excluded.

Question 9: (B) 2 marks

The function receives a mutable reference to the caller's integer. Dereferencing score accesses that integer, so adding 5 updates the caller's value from 10 to 15. *Book: 4.2.*

Question 10: (B) 2 marks

The local String is dropped when label returns. Returning a reference to its contents would leave a dangling reference, so Rust rejects the program. The reference does not own the String or keep its contents alive. *Book: 4.2.*

Question 11: (C) 2 marks

Tuple structs introduce distinct named types, even when their fields have identical types. A Score value cannot be assigned directly to a binding declared as Level, so this assignment fails during compilation. *Book: 5.1.*

Question 12: (A) 2 marks

An associated function without a self parameter is called through its type using double-colon syntax. Limit::new(12) constructs a Limit whose maximum field is 12; no existing instance is required. *Book: 5.3.*

Question 13: (B) 2 marks

Option wraps a potentially present value and is a different type from its contained integer. The integer must first be extracted by matching Some, rather than adding the Option directly to another integer. *Book: 6.1.*

Question 14: (C) 2 marks

Some(3) is a pattern that requires the contained value to be exactly 3. The value Some(4) does not match, so execution takes the else branch and prints other. *Book: 6.3.*

Question 15: (B) 2 marks

The loop starts with count equal to 1, then increments it to 3 and to 5. At 5 the condition count < 5 is false, so the loop ends and prints 5. *Book: 3.5.*

Question 16: (A) 2 marks

Neither explicit integer pattern matches 7, so the catch-all underscore arm runs. The underscore matches any remaining value and makes this match exhaustive without binding the matched value to a name. *Book: 6.2.*

UNIVERSITY OF TORONTO
FACULTY OF APPLIED SCIENCE AND ENGINEERING

ECE 1724-F1 — Performant Software Systems with Rust

Sample Midterm 3 — Answer Key (continued)

Coverage: Rust Book Chapters 1–6, 8, and 9. Chapter 7 is excluded.

Question 17: (D) 2 marks

Both Add(3) and Reset are values of the same enum type, Change. A vector requires one element type but can hold different variants of that enum, so this vector has length 2. *Book: 8.1; 6.1.*

Question 18: (A) 2 marks

String addition takes ownership of the left operand and borrows the right operand. Here first is moved into the addition, so printing first afterward attempts to use a moved value and fails to compile. *Book: 8.2.*

Question 19: (B) 2 marks

The owned String key is moved into the hash map by insert. Because String does not implement Copy, name can no longer be used afterward; the final print therefore causes a compilation error. *Book: 8.3; 4.1.*

Question 20: (D) 2 marks

The question-mark operator propagates an error by returning it from the current function immediately, converting its error type when supported. On Ok, it instead yields the successful value so the function can continue. *Book: 9.2.*

Question 21: (C) 3 marks

The strings 6, 2, and 5 parse successfully, while bad increments rejected once. Only scores at least 5 contribute, so total is 6 + 5 = 11 and rejected is 1. *Book: 2; 3.5; 9.2.*

Question 22: (A) 3 marks

Starting from 1, round 2 becomes 3 and doubles score to 2; round 5 becomes 6 and increases it to 8. Round 1 is skipped, round 4 doubles score to 16, and final shadowing gives 13. *Book: 3.1; 3.5.*

Question 23: (C) 3 marks

Cloning preserves run in backup. rename takes the original String and returns run-new, which the new label binding owns. The integer score is copied into add_point, leaving score at 4 while next receives 5. *Book: 4.1.*

Question 24: (D) 3 marks

The immutable references first and second are last used before the mutable borrow begins. The function appends through a mutable borrow, then label is appended to directly and borrowed again by last. The two printed lines show the original and updated contents. *Book: 4.2.*

Coverage: Rust Book Chapters 1–6, 8, and 9. Chapter 7 is excluded.

Question 25: (B) 3 marks

first_tag returns the slice before the colon, which is gold with four ASCII bytes. tag is last used by the first print, so the later mutation is allowed; clear and push_str leave label containing new. *Book: 4.3; 8.2.*

Question 26: (C) 3 marks

The update syntax supplies only reserved from first because name and count are explicitly set. reserved is a Copy boolean, so first.name and first.count remain usable. The printed values are pens, 8, and 10. *Book: 5.1; 4.1.*

Question 27: (C) 3 marks

new creates a score of 2, so before stores 4. Adding 3 raises the stored score to 5; adding before raises it to 9. The last doubled call returns 18 without changing before. *Book: 5.3.*

Question 28: (B) 3 marks

Gain(4) changes score from 2 to 6, and Reset then sets it to 1. The inactive bonus adds nothing. The final Gain(3) produces 4, demonstrating how each enum payload affects the match arm. *Book: 6.1–6.2.*

Question 29: (A) 3 marks

award transforms Some(3) into Some(7) but leaves None as None. The loop therefore adds 7, subtracts 1, and adds 2. The resulting total is 8; the extra 8 cannot create a missing score. *Book: 6.1–6.2.*

Question 30: (A) 3 marks

The three updates take score from 5 to 9, then to 7, then to 8. The final if let succeeds with extra equal to 3, so score is multiplied by 3 and prints 24. *Book: 6.3; 4.2.*

Question 31: (A) 3 marks

value only borrows the Score, but finish takes self by value and consumes game. The later call to game.value() therefore uses a moved value. Compilation fails, so even the earlier print does not run. *Book: 5.3; 4.1.*

Question 32: (D) 3 marks

The first three rounds add 3 to the initial 1, double 4 to 8, and add 2 to reach 10. End breaks before played is incremented, so played is 3 and the final Points(9) is skipped. *Book: 6.2; 3.5.*

Coverage: Rust Book Chapters 1–6, 8, and 9. Chapter 7 is excluded.

Question 33: (B) 3 marks

Successful parses append 4 and 3 to the initial vector [2]. Parsing bad returns early through the question-mark operator without appending, and main counts one rejection. The final vector has three elements totaling 9. *Book: 8.1; 9.2; 4.2.*

Question 34: (C) 3 marks

Mutation changes the vector to [4, 7, 3]. first is last used before push, so the append is allowed and adds 5. Iteration sums [4, 7, 3, 5] to 19; index 3 contains 5. *Book: 8.1; 4.2.*

Question 35: (C) 3 marks

The vector stores variants of one enum type. Add(5) changes count from 3 to 8, Remove(2) makes it 6, and Add(4) makes it 10. Ignore changes neither value, so three changes are counted as applied. *Book: 8.1; 6.2.*

Question 36: (D) 3 marks

format! borrows its arguments, and addition moves label while borrowing suffix. Thus team, stage, and suffix remain usable. The resulting ASCII text red:one!end contains three e bytes, one in each of red, one, and end. *Book: 8.2; 4.1.*

Question 37: (C) 3 marks

Deliveries raise pens from 2 to 6 and create clips with 4. The later or_insert(10) preserves the existing clips value, then subtraction makes it 3. Tape is inserted with 2, giving a total of 11. *Book: 8.3.*

Question 38: (D) 3 marks

The first call prints first 4, then returns an error when bad fails to parse; main subtracts 1. The second call prints first 2 and returns 5, so the final score is -1 + 5 = 4. *Book: 9.2.*

Question 39: (D) 3 marks

The first update parses as 5 and raises stock from 4 to 9. The invalid input is handled by subtracting 2, leaving 7. The final signed integer is -1, so adding it produces 6. *Book: 9.2; 2.*

Question 40: (A) 3 marks

The first parse error is recovered through match, setting score to 3. The valid bonus raises it to 10. The final expect receives Err and panics, so the next print is never executed. *Book: 9.1–9.2.*

UNIVERSITY OF TORONTO
FACULTY OF APPLIED SCIENCE AND ENGINEERING

ECE 1724-F1 — Performant Software Systems with Rust

Sample Midterm 3 — Coverage and Timing

Coverage: Rust Book Chapters 1–6, 8, and 9. Chapter 7 is excluded.

Scoring and pacing

Section	Questions	Marks	Estimated time
A: Easy	20	40	40 minutes
B: Harder code reading	20	60	66 minutes 40 seconds
Review	—	—	3 minutes 20 seconds
Total	40	100	110 minutes

Easy pacing: five questions per ten minutes. Harder pacing: three questions per ten minutes. Harder questions combine familiar concepts in longer code; they do not require material beyond Chapters 1–6, 8, and 9.

Coverage map and official sources

Rust Book chapter	Questions
1: Getting Started	1, 2
2: Programming a Guessing Game	3, 4, 21, 39
3: Common Programming Concepts	5–7, 15, 21, 22, 32
4: Understanding Ownership	8–10, 23–26, 30, 31, 33, 34, 36
5: Using Structs	11, 12, 26, 27, 31
6: Enums and Pattern Matching	13, 14, 16, 28–30, 32, 35
8: Common Collections	17–19, 25, 33–37
9: Error Handling	20, 21, 33, 38–40

Source: The Rust Programming Language, Chapters 1–6, 8, and 9. Individual answer explanations identify the relevant sections. Official chapter links:

1: Getting Started 2: Guessing Game

3: Common Concepts 4: Ownership

5: Structs 6: Enums

8: Collections 9: Error Handling