UNIVERSITY OF TORONTO
FACULTY OF APPLIED SCIENCE AND ENGINEERING

ECE 1724-F1 — Performant Software Systems with Rust
Sample Midterm Examination — Set 3
110 minutes / 100 marks

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

This practice examination contains 40 multiple-choice questions. Select exactly one answer (A–D) for each question. There is no negative marking.

Suggested pacing: Allow 40 minutes for Questions 1–20 (2 marks each), approximately 67 minutes for Questions 21–40 (3 marks each), and 3 minutes to review your answers.

The harder questions assess your ability to read longer Rust programs using the same foundational concepts. All questions are independent.

The examination has 18 pages, including this cover.

First Name: _____ Last Name: _____

Student Number: _____

MARKS

Questions 1–20	Questions 21–40	Total
/40	/60	/100

Duration: 110 minutes **Total: 100 marks**

Choose exactly one answer (A–D) for each question. There is no negative marking. Each question is independent. Unless a complete program is shown, treat code as a snippet inside `main`. Compilation failures print nothing. Ignore compiler warnings. Assume Rust 2024. For timed practice, work without a compiler and check solutions afterward.

Section A: 20 easy questions, 2 marks each (40 marks; about 40 minutes). Section B: 20 harder code-reading questions, 3 marks each (60 marks; about 67 minutes). Reserve about 3 minutes to review.

Section A — Easy questions

Question 1 [2 Marks]

Which file normally contains a Cargo package's name, version, and declared dependencies?

(A) Cargo.lock
(B) target/debug
(C) src/main.rs
(D) Cargo.toml

Question 2 [2 Marks]

After rustc main.rs succeeds, what has normally happened?

(A) The program has already run.
(B) A new Cargo package has been created.
(C) An executable has been created; running it is a separate step.
(D) The source file has been replaced by machine code.

Question 3 [2 Marks]

In the guessing-game input pattern, why is expect called after read_line?

(A) To compare the guess with the secret number.
(B) To convert the entered text into an integer.
(C) To remove the newline from the input.
(D) To handle the Result returned by the input operation.

Sample Midterm 3 — Questions (continued)

Question 4 [2 Marks]

Inside a loop, a parse result is matched with an Err(_) => continue arm. What happens when parsing fails?

(A) The entire program exits.

(B) The remaining loop body is skipped and the next iteration begins.

(C) The failed input becomes zero.

(D) The loop is exited permanently.

Question 5 [2 Marks]

Which statement about a Rust const declaration is correct?

(A) Its type must be written explicitly, and its value cannot be reassigned.

(B) Its value can be changed if it is declared with mut.

(C) It must be declared inside main.

(D) It always owns heap memory.

Question 6 [2 Marks]

Which statement correctly contrasts arrays and tuples?

(A) Array elements have one type; tuple fields can have different types.

(B) Tuple fields must all have one type; array elements can differ.

(C) Both arrays and tuples must contain only integers.

(D) An array grows automatically when assigned another element.

Sample Midterm 3 — Questions (continued)

Question 7 [2 Marks]

Which statement describes this code?

```
let active = true;
let status = if active { 3 } else { "off" };
println!("{}", status);
```

(A) It prints 3.

(B) It fails to compile because the if branches have incompatible types.

(C) It panics when active is false.

(D) It prints off.

Question 8 [2 Marks]

What does this code print?

```
let original = String::from("go");
let mut copy = original.clone();
copy.push_str("al");
println!("{} {}", original, copy);
```

(A) goal goal

(B) go go

(C) The code does not compile.

(D) go goal

Question 9 [2 Marks]

What does this complete program print?

```
fn boost(score: &mut i32) {
    *score += 5;
}
fn main() {
    let mut score = 10;
    boost(&mut score);
    println!("{}", score);
}
```

(A) 10

(B) 15

(C) 5

(D) The code does not compile.

Question 10 [2 Marks]

Which statement describes this complete program?

```
fn label() -> &String {
    let text = String::from("team");
    &text
}
fn main() {
    println!("{}", label());
}
```

(A) It prints team.

(B) It fails because the returned reference would refer to a local String.

(C) It prints an unpredictable value.

(D) It prints an empty string.

Question 11 [2 Marks]

Which statement describes this code?

```
struct Score(i32);
struct Level(i32);
let score = Score(4);
let level: Level = score;
println!("{}", level.0);
```

(A) It prints 4.

(B) It prints 0.

(C) It fails because Score and Level are different types.

(D) It compiles but panics during the assignment.

Question 12 [2 Marks]

Which call correctly invokes the associated function below to make a value?

```
struct Limit { maximum: u32 }
impl Limit {
    fn new(maximum: u32) -> Limit {
        Limit { maximum }
    }
}
```

(A) Limit::new(12)

(B) new::Limit(12)

(C) Limit.new(12)

(D) Limit->new(12)

Question 13 [2 Marks]

Which statement describes this code?

```
let bonus = Some(4);
let total = 3 + bonus;
println!("{}", total);
```

(A) It prints 7.

(B) It fails because an integer cannot be added directly to Option<i32>.

(C) It panics because Some is not None.

(D) It prints 3.

Question 14 [2 Marks]

What does this code print?

```
let level = Some(4);
if let Some(3) = level {
    println!("three");
} else {
    println!("other");
}
```

(A) The code does not compile.

(B) three

(C) other

(D) It prints nothing.

Question 15 [2 Marks]

What does this code print?

```
let mut count = 1;
while count < 5 {
    count += 2;
}
println!("{}", count);
```

(A) 3

(B) 5

(C) 4

(D) 7

Question 16 [2 Marks]

What does this code print?

```
let level = 7;
match level {
    1 => println!("start"),
    5 => println!("middle"),
    _ => println!("other"),
}
```

(A) other

(B) middle

(C) 7

(D) The code does not compile.

Question 17 [2 Marks]

Which statement describes this code?

```
enum Change { Add(i32), Reset }
let changes = vec![Change::Add(3), Change::Reset];
println!("{}", changes.len());
```

(A) It panics when constructing the vector.

(B) It fails because the variants have different payloads.

(C) It prints 3.

(D) It prints 2; both variants have type Change.

Question 18 [2 Marks]

Which statement describes this code?

```
let first = String::from("red");
let second = String::from("team");
let label = first + &second;
println!("{} {}", first, label);
```

(A) It fails because + moves the left-hand String.

(B) It prints redteam redteam.

(C) It panics because second is borrowed.

(D) It prints red redteam.

Question 19 [2 Marks]

Which statement describes this code?

```
use std::collections::HashMap;
let mut stock = HashMap::new();
let name = String::from("pens");
stock.insert(name, 4);
println!("{}", name);
```

(A) It prints pens.

(B) It fails because inserting the owned String moves name.

(C) It prints 4.

(D) It panics because the map has no keys.

Question 20 [2 Marks]

In a function returning Result, what does applying ? to an Err value normally do?

(A) It always panics.

(B) It skips only the next statement and continues.

(C) It replaces the error with zero.

(D) It returns an error from the current function immediately.

Sample Midterm 3 — Questions (continued)

Section B — Harder code-reading questions

Question 21 [3 Marks]

What does this complete program print?

```
fn main() {
    let inputs = [" 6 ", "bad", "2", " 5"];
    let mut total = 0;
    let mut rejected = 0;
    for input in inputs {
        let parsed = input.trim().parse::<i32>();
        match parsed {
            Ok(points) => {
                if points >= 5 {
                    total += points;
                }
            }
            Err(_) => rejected += 1,
        }
    }
    println!("{} {}", total, rejected);
}
```

(A) 13 1

(B) 11 2

(C) 11 1

(D) 6 1

Question 22 [3 Marks]

What does this complete program print?

```
fn main() {
    let rounds = [2, 5, 1, 4];
    let mut score = 1;
    for points in rounds {
        if points < 2 {
            continue;
        }
        let points = points + 1;
        if points % 2 == 0 {
            score += points;
        } else {
            score *= 2;
        }
    }
    let score = score - 3;
    println!("{}", score);
}
```

(A) 13

(B) 16

(C) 9

(D) The code does not compile because points is immutable.

Question 23 [3 Marks]

What does this complete program print?

```
fn rename(mut label: String) -> String {
    label.push_str("-new");
    label
}
fn add_point(mut score: i32) -> i32 {
    score += 1;
    score
}
fn main() {
    let label = String::from("run");
    let backup = label.clone();
    let label = rename(label);
    let score = 4;
    let next = add_point(score);
    println!("{} {} {} {}", backup, label, score, next);
}
```

(A) run-new run-new 5 5

(B) run run-new 5 5

(C) run run-new 4 5

(D) The code does not compile because label is moved.

Question 24 [3 Marks]

Which statement describes this complete program?

```
fn add_round(label: &mut String) {
    label.push_str("!");
}
fn main() {
    let mut label = String::from("round");
    let first = &label;
    let second = &label;
    println!("{} {}", first, second);
    add_round(&mut label);
    label.push_str("?");
    let last = &label;
    println!("{}", last);
}
```

(A) It fails because any immutable borrow prevents later mutation.

(B) It fails because add_round does not return the String.

(C) It prints round round, then round on the next line.

(D) It prints round round, then round!? on the next line.

Question 25 [3 Marks]

What does this complete program print? All characters are ASCII.

```
fn first_tag(text: &str) -> &str {
    let bytes = text.as_bytes();
    for (index, &byte) in bytes.iter().enumerate() {
        if byte == b':' {
            return &text[..index];
        }
    }
    text
}
fn main() {
    let mut label = String::from("gold:team");
    let tag = first_tag(&label);
    let length = tag.len();
    println!("{} {}", tag, length);
    label.clear();
    label.push_str("new");
    println!("{}", label);
}
```

(A) gold:team 9, then new on the next line.

(B) gold 4, then new on the next line.

(C) It fails because tag prevents clear even after its last use.

(D) gold 4, then an empty line.

Question 26 [3 Marks]

What does this complete program print?

```
struct Stock {
    name: String,
    count: u32,
    reserved: bool,
}
fn main() {
    let first = Stock {
        name: String::from("pens"),
        count: 8,
        reserved: false,
    };
    let second = Stock {
        name: String::from("clips"),
        count: first.count + 2,
        ..first
    };
    println!("{} {} {}", first.name, first.count, second.count);
}
```

(A) It fails because ..first always moves every field.

(B) clips 8 10

(C) pens 8 10

(D) pens 10 10

Question 27 [3 Marks]

What does this complete program print?

```
struct Score { points: i32 }
impl Score {
    fn new() -> Score { Score { points: 2 } }
    fn add(&mut self, amount: i32) {
        self.points += amount;
    }
    fn doubled(&self) -> i32 {
        self.points * 2
    }
}
fn main() {
    let mut game = Score::new();
    let before = game.doubled();
    game.add(3);
    game.add(before);
    println!("{} {}", before, game.doubled());
}
```

(A) 4 10

(B) 10 18

(C) 4 18

(D) The code does not compile.

Question 28 [3 Marks]

What does this complete program print?

```
enum Event { Gain(i32), Reset, Bonus { amount: i32, active: bool } }
fn main() {
    let events = [Event::Gain(4), Event::Reset,
        Event::Bonus { amount: 6, active: false },
        Event::Gain(3)];
    let mut score = 2;
    for event in events {
        match event {
            Event::Gain(points) => score += points,
            Event::Reset => score = 1,
            Event::Bonus { amount, active } => {
                if active { score += amount; }
            }
        }
    }
    println!("{}", score);
}
```

(A) 10

(B) 4

(C) 3

(D) 9

Question 29 [3 Marks]

What does this complete program print?

```
fn award(current: Option<i32>, extra: i32) -> Option<i32> {
    match current {
        Some(points) => Some(points + extra),
        None => None,
    }
}
fn main() {
    let first = award(Some(3), 4);
    let second = award(None, 8);
    let mut total = 0;
    for score in [first, second, Some(2)] {
        match score {
            Some(points) => total += points,
            None => total -= 1,
        }
    }
    println!("{}", total);
}
```

(A) 8

(B) 9

(C) 17

(D) The code does not compile.

Question 30 [3 Marks]

What does this complete program print?

```
fn update(current: &mut i32, bonus: Option<i32>) {
    if let Some(points) = bonus {
        *current += points;
    } else {
        *current -= 2;
    }
}
fn main() {
    let bonuses = [Some(4), None, Some(1)];
    let mut score = 5;
    for bonus in bonuses {
        update(&mut score, bonus);
    }
    if let Some(extra) = Some(3) {
        score *= extra;
    }
    println!("{}", score);
}
```

(A) 24

(B) 30

(C) 15

(D) 18

Question 31 [3 Marks]

Which statement describes this complete program?

```
struct Score { points: i32 }
impl Score {
    fn value(&self) -> i32 {
        self.points
    }
    fn finish(self) -> i32 {
        self.points + 5
    }
}
fn main() {
    let game = Score { points: 7 };
    let before = game.value();
    let final_score = game.finish();
    println!("{} {}", before, final_score);
    println!("{}", game.value());
}
```

(A) It fails to compile because finish consumes game.

(B) It prints 7 12 before a runtime error.

(C) It prints 7 12, then 12 on the next line.

(D) It prints 7 12, then 7 on the next line.

Question 32 [3 Marks]

What does this complete program print?

```
enum Round {
    Points(i32),
    Double,
    End,
}
fn main() {
    let rounds = [Round::Points(3), Round::Double,
        Round::Points(2), Round::End, Round::Points(9)];
    let mut score = 1;
    let mut played = 0;
    for round in rounds {
        match round {
            Round::Points(value) => score += value,
            Round::Double => score *= 2,
            Round::End => break,
        }
        played += 1;
    }
    println!("{} {}", score, played);
}
```

(A) 19 5

(B) 19 4

(C) 10 4

(D) 10 3

Question 33 [3 Marks]

What does this complete program print?

```
fn add_stock(stock: &mut Vec<i32>, text: &str)
    -> Result<(), std::num::ParseIntError>
{
    let count = text.parse::<i32>()?;
    stock.push(count);
    Ok(())
}
fn main() {
    let mut stock = vec![2];
    let mut rejected = 0;
    for text in ["4", "bad", "3"] {
        match add_stock(&mut stock, text) {
            Ok(()) => (),
            Err(_) => rejected += 1,
        }
    }
    let mut total = 0;
    for count in &stock { total += *count; }
    println!("{} {} {}", stock.len(), total, rejected);
}
```

(A) 4 9 1

(B) 3 9 1

(C) 3 7 1

(D) It panics on bad.

Question 34 [3 Marks]

What does this complete program print?

```
fn main() {
    let mut stock = vec![2, 5, 1];
    for count in &mut stock {
        *count += 2;
    }
    let first = &stock[0];
    let saved = *first;
    println!("{}", first);
    stock.push(saved + 1);
    let mut total = 0;
    for count in &stock {
        total += *count;
    }
    let last = stock[3];
    println!("{} {}", total, last);
}
```

(A) 4, then 14 3 on the next line.

(B) It fails because first prevents push after its last use.

(C) 4, then 19 5 on the next line.

(D) 2, then 11 3 on the next line.

Question 35 [3 Marks]

What does this complete program print?

```
enum Change {
    Add(i32),
    Remove(i32),
    Ignore,
}
fn main() {
    let changes = vec![Change::Add(5), Change::Remove(2),
        Change::Ignore, Change::Add(4)];
    let mut count = 3;
    let mut applied = 0;
    for change in changes {
        match change {
            Change::Add(value) => { count += value; applied += 1; }
            Change::Remove(value) => { count -= value; applied += 1; }
            Change::Ignore => (),
        }
    }
    println!("{} {}", count, applied);
}
```

(A) 7 3

(B) 10 4

(C) 10 3

(D) The code does not compile.

Question 36 [3 Marks]

What does this complete program print?

```
fn main() {
    let team = String::from("red");
    let stage = String::from("one");
    let mut label = format!("{}:{}", team, stage);
    label.push('!');
    let suffix = String::from("end");
    let final_label = label + &suffix;
    let mut count = 0;
    for byte in final_label.bytes() {
        if byte == b'e' {
            count += 1;
        }
    }
    println!("{} {} {}", team, stage, suffix);
    println!("{} {}", final_label, count);
}
```

(A) It fails because format! moves team and stage.

(B) It fails because + moves suffix.

(C) red one end, then red:one!end 2 on the next line.

(D) red one end, then red:one!end 3 on the next line.

Question 37 [3 Marks]

What does this complete program print?

```
use std::collections::HashMap;
fn main() {
    let mut stock = HashMap::new();
    stock.insert("pens", 2);
    let deliveries = [("pens", 3), ("clips", 4), ("pens", 1)];
    for (name, amount) in deliveries {
        let count = stock.entry(name).or_insert(0);
        *count += amount;
    }
    let clips = stock.entry("clips").or_insert(10);
    *clips -= 1;
    stock.entry("tape").or_insert(2);
    let total = *stock.get("pens").unwrap()
        + *stock.get("clips").unwrap() + *stock.get("tape").unwrap();
    println!("{} {} {}", stock.get("pens").unwrap(),
        stock.get("clips").unwrap(), total);
}
```

(A) 6 13 21

(B) 4 3 9

(C) 6 3 11

(D) 6 4 12

Question 38 [3 Marks]

What does this complete program print?

```
fn total(first: &str, second: &str) -> Result<i32, std::num::ParseIntError> {
    let a = first.parse::<i32>()?;
    println!("first {}", a);
    let b = second.parse::<i32>()?;
    Ok(a + b)
}
fn main() {
    let inputs = [("4", "bad"), ("2", "3")];
    let mut score = 0;
    for (first, second) in inputs {
        match total(first, second) {
            Ok(value) => score += value,
            Err(_) => score -= 1,
        }
    }
    println!("score {}", score);
}
```

(A) first 4, first 2, score 9 on three lines.

(B) first 2, score 5 on two lines.

(C) first 4, then it panics.

(D) first 4, first 2, score 4 on three lines.

Question 39 [3 Marks]

What does this complete program print?

```
fn quantity(text: &str) -> Result<i32, std::num::ParseIntError> {
    text.trim().parse::<i32>()
}
fn apply(current: i32, text: &str) -> i32 {
    match quantity(text) {
        Ok(value) => current + value,
        Err(_) => current - 2,
    }
}
fn main() {
    let updates = [" 5 ", "none", "-1"];
    let mut stock = 4;
    for update in updates {
        stock = apply(stock, update);
    }
    println!("{}", stock);
}
```

(A) 8

(B) 9

(C) It panics on none.

(D) 6

Question 40 [3 Marks]

Which statement describes this complete program? Ignore the text of any panic message.

```
fn read_score(text: &str) -> Result<i32, std::num::ParseIntError> {
    text.parse::<i32>()
}
fn main() {
    let first = read_score("bad");
    let mut score = match first {
        Ok(value) => value,
        Err(_) => 3,
    };
    println!("start {}", score);
    let bonus = read_score("7").expect("valid bonus");
    score += bonus;
    println!("score {}", score);
    let next = read_score("oops").expect("valid next score");
    println!("next {}", next);
}
```

(A) It prints start 3 and score 10, then panics.

(B) It panics before printing any lines.

(C) It prints start 3, score 10, and next 0.

(D) It fails to compile because expect cannot handle Result.