UNIVERSITY OF TORONTO
FACULTY OF APPLIED SCIENCE AND ENGINEERING

ECE 1724-F1 — Performant Software Systems with Rust
Sample Midterm Examination — Set 2
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. Unless a complete program is shown, code is inside `main`. Programs that fail compilation print nothing. Ignore compiler warnings; assume Rust 2024. For timed practice, answer without a compiler, then check solutions afterward.

The examination has 16 pages, including this cover.

First Name: _____ Last Name: _____

Student Number: _____

MARKS

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

Section A: Easy questions — 20 questions, 2 marks each (40 marks).

Question 1 [2 Marks]

Which Cargo command builds an optimized release executable?

(A) cargo build --release
(B) cargo run
(C) cargo check
(D) cargo new --release

Question 2 [2 Marks]

Which statement correctly distinguishes cargo run from cargo build?

(A) cargo run only checks syntax.
(B) cargo build runs the executable after building.
(C) Both commands always run the executable.
(D) cargo run builds and runs; cargo build builds without running.

Question 3 [2 Marks]

What happens when this code runs? The string includes surrounding spaces.

```
let line = " 24 ";
let result = line.parse::<u32>();
match result {
    Ok(n) => println!("{}", n),
    Err(_) => println!("invalid"),
}
```

(A) It prints 24.
(B) It prints invalid.
(C) It panics.
(D) It fails to compile.

Question 4 [2 Marks]

What does this code print?

```
use std::cmp::Ordering;
let measured = 8;
let target = 10;
match measured.cmp(&target) {
    Ordering::Less => println!("low"),
    Ordering::Equal => println!("equal"),
    Ordering::Greater => println!("high"),
}
```

(A) equal
(B) high
(C) A compilation error
(D) low

Sample Midterm 2 — Questions (continued)

Question 5 [2 Marks]

What happens when this code is compiled and run?

```
let mut reading = 5;
reading = "five";
println!("{}", reading);
```

(A) It prints five.

(B) It compiles, then panics.

(C) It prints 5.

(D) It fails to compile.

Question 6 [2 Marks]

What does this code print?

```
let readings = [6; 3];
println!("{} {}", readings[1], readings.len());
```

(A) 3 6

(B) 6 6

(C) 3 3

(D) 6 3

Question 7 [2 Marks]

What happens when this complete program is compiled and run?

```
fn adjusted(value: i32) -> i32 {
    value + 3;
}
fn main() {
    println!("{}", adjusted(6));
}
```

(A) It fails to compile.

(B) It prints 9.

(C) It prints 6.

(D) It compiles, then panics.

Question 8 [2 Marks]

What does this code print?

```
let station = String::from("north");
let saved = station;
let final_name = saved;
println!("{}", final_name);
```

(A) It fails because a String cannot move twice.

(B) It prints north.

(C) It prints an empty line.

(D) It panics.

Question 9 [2 Marks]

What does this complete program print?

```
fn increase(mut count: i32) {
    count += 1;
    println!("{}", count);
}
fn main() {
    let count = 7;
    increase(count);
    println!("{}", count);
}
```

(A) A compilation error
(B) 8 then 8
(C) 7 then 7
(D) 8 then 7

Question 10 [2 Marks]

What does this code print? All characters here are ASCII.

```
let code = String::from("RAIN-42");
let suffix = &code[5..];
println!("{}", suffix);
```

(A) -42
(B) RAIN
(C) 2
(D) 42

Question 11 [2 Marks]

What does this code print?

```
struct Reading { value: i32, valid: bool }
let value = 12;
let valid = false;
let reading = Reading { value, valid };
println!("{} {}", reading.value, reading.valid);
```

(A) 12 false
(B) 0 false
(C) 12 true
(D) A compilation error

Question 12 [2 Marks]

What happens when this code is compiled and run?

```
#[derive(Debug)]
struct Reading { value: i32 }
let reading = Reading { value: 12 };
println!("{}", reading);
```

(A) It prints Reading { value: 12 }.
(B) It prints 12.
(C) It fails because {} requires Display.
(D) It panics because Debug cannot print structs.

Question 13 [2 Marks]

What does this code print?

```
enum Weather { Dry, Rain(u32) }
let weather = Weather::Rain(3);
let amount = match weather {
    Weather::Dry => 0,
    Weather::Rain(mm) => mm * 2,
};
println!("{}", amount);
```

(A) 0

(B) 6

(C) A compilation error

(D) 3

Question 14 [2 Marks]

What happens when this code is compiled and run?

```
let reading: Option<i32> = Some(4);
let value = match reading {
    Some(n) => n + 1,
};
println!("{}", value);
```

(A) It prints 4.

(B) It prints 5.

(C) It fails because None is not covered.

(D) It compiles, then panics.

Question 15 [2 Marks]

What does this code print?

```
let forecast = [3, 6, 9];
let (first, last) = (forecast[0], forecast[2]);
println!("{} {}", first + last, forecast.len());
```

(A) 9 2

(B) 12 2

(C) 12 3

(D) 18 3

Question 16 [2 Marks]

What does this code print?

```
let mut count = 1;
let reading = Some(4);
if let Some(value) = reading {
    count += value;
}
println!("{}", count);
```

(A) 5

(B) 1

(C) 4

(D) A compilation error

Sample Midterm 2 — Questions (continued)

Question 17 [2 Marks]

What happens when this code runs?

```
let readings = vec![3, 8];
let value = readings[2];
println!("{}", value);
```

(A) It fails to compile.
(B) It prints 8.
(C) It prints 0.
(D) It panics before printing.

Question 18 [2 Marks]

The first character in this string uses two UTF-8 bytes; each ASCII character uses one. What does this code print?

```
let label = String::from("\u{00e9}ast");
println!("{}", label.len());
```

(A) 3
(B) 4
(C) It panics.
(D) 5

Question 19 [2 Marks]

What does this code print?

```
use std::collections::HashMap;
let mut counts = HashMap::new();
counts.insert("rain", 2);
counts.entry("rain").or_insert(5);
println!("{}", *counts.get("rain").unwrap());
```

(A) 5
(B) 7
(C) 2
(D) It panics.

Question 20 [2 Marks]

What happens when this code runs?

```
let result = "rain".parse::<u32>();
let count = result.expect("number required");
println!("{}", count);
```

(A) It prints 0.
(B) It panics before printing.
(C) It prints number required.
(D) It fails to compile.

Sample Midterm 2 — Questions (continued)

Section B: Longer code reading — 20 questions, 3 marks each (60 marks).

Question 21 [3 Marks]

What does this code print?

```
let lines = [" 8 ", "3", "bad", "12"];
let mut total = 0;
let mut skipped = 0;
for line in lines {
    let value: i32 = match line.trim().parse() {
        Ok(n) => n,
        Err(_) => {
            skipped += 1;
            break;
        }
    };
    if value >= 5 {
        total += value;
    } else {
        total += 1;
    }
}
println!("{} {}", total, skipped);
```

(A) 9 1

(B) 21 1

(C) 8 2

(D) 9 0

Question 22 [3 Marks]

What does this code print?

```
let levels = [2, 5, 1, 4];
let mut total = 0;
let mut used = 0;
for level in levels {
    let level = level + 1;
    if level % 2 == 0 {
        total += level;
    } else {
        total += 1;
    }
    used += 1;
    if total > 7 {
        break;
    }
}
println!("{} {}", total, used);
```

(A) 9 3

(B) 7 2

(C) 8 3

(D) 10 4

Question 23 [3 Marks]

What does this complete program print?

```
fn tag(mut name: String) -> String {
    name.push_str("-ok");
    name
}
fn main() {
    let mut station = String::from("bay");
    let backup = station.clone();
    station.push_str("-2");
    let tagged = tag(station);
    let mut display = backup;
    display.push('!');
    println!("{}", tagged);
    println!("{}", display);
}
```

(A) bay-2-ok then bay-2!

(B) bay-ok then bay-2!

(C) bay-2-ok then bay!

(D) A compilation error

Question 24 [3 Marks]

What happens when this complete program is compiled and run?

```
fn add_marker(text: &mut String) {
    text.push('!');
}
fn width(text: &String) -> usize {
    text.len()
}
fn main() {
    let mut label = String::from("dock");
    let view = &label;
    let before = width(view);
    println!("{}", view);
    add_marker(&mut label);
    let after = width(&label);
    println!("{} {}", before, after);
}
```

(A) It fails because view remains in scope.

(B) It prints dock! then 4 5.

(C) It prints dock then 4 5.

(D) It panics during add_marker.

Question 25 [3 Marks]

What does this code print? All characters here are ASCII.

```
let mut report = String::from("sun wind");
let mut split = report.len();
for (index, byte) in report.as_bytes().iter().enumerate() {
    if *byte == b' ' {
        split = index;
        break;
    }
}
let first = &report[..split];
println!("{}", first);
let saved = String::from(first);
report.clear();
report.push_str("rain");
println!("{} {}", saved, report);
```

(A) sun then rain rain

(B) sun wind then sun wind rain

(C) sun then sun rain

(D) A compilation error

Question 26 [3 Marks]

What happens when this code is compiled and run?

```
struct Shipment {
    destination: String,
    boxes: u32,
    urgent: bool,
}
let original = Shipment {
    destination: String::from("west"),
    boxes: 4,
    urgent: false,
};
let revised = Shipment {
    destination: String::from("south"),
    boxes: 7,
    ..original
};
println!("{} {}", original.boxes, revised.boxes);
println!("{}", original.destination);
```

(A) It prints 4 7, then south.

(B) It prints 7 7, then west.

(C) It fails because destination was moved.

(D) It prints 4 7, then west.

Question 27 [3 Marks]

What does this complete program print?

```
struct Gauge { value: i32 }
impl Gauge {
    fn new(value: i32) -> Self {
        Self { value }
    }
    fn raise(&mut self, amount: i32) {
        self.value += amount;
    }
    fn above(&self, limit: i32) -> bool {
        self.value > limit
    }
}
fn main() {
    let mut gauge = Gauge::new(5);
    gauge.raise(4);
    let high = gauge.above(8);
    gauge.raise(-2);
    println!("{} {}", gauge.value, high);
}
```

(A) 7 false

(B) 7 true

(C) 9 true

(D) A compilation error

Question 28 [3 Marks]

What does this complete program print?

```
enum Event {
    Warm(i32),
    Rain(i32),
    Clear,
}
fn points(event: Event) -> i32 {
    match event {
        Event::Warm(n) => n * 2,
        Event::Rain(n) => n + 1,
        Event::Clear => 0,
    }
}
fn main() {
    let first = Event::Warm(3);
    let second = Event::Rain(4);
    let mut total = points(first);
    total += points(second);
    total += points(Event::Clear);
    println!("{}", total);
}
```

(A) 7

(B) 11

(C) 10

(D) 12

Question 29 [3 Marks]

What does this complete program print?

```
fn adjust(value: Option<i32>) -> Option<i32> {
    match value {
        Some(n) => {
            if n > 0 {
                Some(n + 2)
            } else {
                None
            }
        }
        None => None,
    }
}
fn main() {
    let first = adjust(Some(4));
    let second = adjust(Some(0));
    let a = match first { Some(n) => n, None => 10 };
    let b = match second { Some(n) => n, None => 10 };
    println!("{} {}", a, b);
}
```

(A) 6 2

(B) 6 10

(C) 6 0

(D) 4 0

Question 30 [3 Marks]

What does this complete program print?

```
fn record(reading: Option<i32>, total: &mut i32) {
    let Some(value) = reading else {
        *total += 1;
        return;
    };
    if value >= 5 {
        *total += value;
    } else {
        *total += 2;
    }
}
fn main() {
    let mut total = 0;
    record(Some(7), &mut total);
    record(None, &mut total);
    record(Some(3), &mut total);
    println!("{}", total);
}
```

(A) 10

(B) 11

(C) 12

(D) A compilation error

Question 31 [3 Marks]

What happens when this complete program is compiled and run?

```
struct Record { label: String }
impl Record {
    fn width(&self) -> usize {
        self.label.len()
    }
    fn into_label(self) -> String {
        self.label
    }
}
fn main() {
    let record = Record { label: String::from("rain") };
    let before = record.width();
    let label = record.into_label();
    let after = record.width();
    println!("{} {} {}", before, after, label);
}
```

(A) It prints 4 4 rain.

(B) It fails because into_label takes ownership of record.

(C) It prints 4 0 rain.

(D) It fails because width takes ownership of record.

Question 32 [3 Marks]

What does this code print?

```
enum Update {
    Add(i32),
    Remove(i32),
    Reset,
}
let updates = [
    Update::Add(5),
    Update::Remove(2),
    Update::Reset,
    Update::Add(7),
];
let mut current = 0;
let mut total = 0;
for update in updates {
    match update {
        Update::Add(n) => current += n,
        Update::Remove(n) => current -= n,
        Update::Reset => current = 0,
    }
    total += current;
}
println!("{} {}", current, total);
```

(A) 7 7

(B) 7 15

(C) 10 15

(D) 0 8

Question 33 [3 Marks]

What does this complete program print?

```
fn add_reading(values: &mut Vec<i32>, text: &str) -> bool {
    match text.parse::<i32>() {
        Ok(value) => {
            values.push(value);
            true
        }
        Err(_) => false,
    }
}
fn main() {
    let mut values = vec![2];
    let mut accepted = 0;
    for text in ["3", "bad", "5"] {
        if add_reading(&mut values, text) {
            accepted += 1;
        }
    }
    let mut total = 0;
    for value in &values {
        total += *value;
    }
    println!("{} {} {}", total, values.len(), accepted);
}
```

(A) 8 2 3

(B) 10 2 2

(C) 10 3 2

(D) It panics while parsing bad.

Question 34 [3 Marks]

What does this code print?

```
let mut readings = vec![2, 4, 6];
for reading in &mut readings {
    *reading += 1;
}
let first = readings[0];
readings.push(first + 2);
let mut total = 0;
for reading in &readings {
    if *reading >= 5 {
        total += *reading;
    } else {
        total += 1;
    }
}
println!("{} {}", total, readings.len());
```

(A) 19 4

(B) 13 3

(C) 18 4

(D) 16 4

Question 35 [3 Marks]

What does this code print?

```
enum Log {
    Loaded(i32),
    Unloaded(i32),
    Note(String),
}
let logs = vec![
    Log::Loaded(8),
    Log::Note(String::from("stop")),
    Log::Unloaded(3),
    Log::Loaded(2),
];
let mut boxes = 0;
let mut notes = 0;
for log in logs {
    match log {
        Log::Loaded(n) => boxes += n,
        Log::Unloaded(n) => boxes -= n,
        Log::Note(text) => notes += text.len(),
    }
}
println!("{} {}", boxes, notes);
```

(A) 13 1

(B) 7 4

(C) 7 1

(D) 13 4

Question 36 [3 Marks]

What does this code print? Semicolons in the choices separate output lines.

```
let mut area = String::from("east");
let suffix = String::from("gate");
let heading = format!("{}:{}", area, suffix);
area.push('!');
let joined = area + "-" + &suffix;
let mut short = suffix.clone();
short.push('2');
let size = joined.len();
let mut shown = String::from("[");
shown.push_str(&short);
shown.push(']');
println!("{}", heading);
println!("{} {}", joined, size);
println!("{} {}", suffix, shown);
```

(A) east!:gate; east!-gate 10; gate [gate2]

(B) east:gate; east-gate 9; gate2 [gate2]

(C) east:gate; east!-gate 10; gate [gate2]

(D) A compilation error

Question 37 [3 Marks]

What does this code print?

```
use std::collections::HashMap;
let events = ["rain", "sun", "rain", "wind"];
let mut counts = HashMap::new();
for event in events {
    let count = counts.entry(event).or_insert(0);
    *count += 1;
}
counts.entry("sun").or_insert(5);
let wind = counts.entry("wind").or_insert(5);
*wind += 2;
counts.insert("rain", 7);
let rain_count = *counts.get("rain").unwrap();
let sun_count = *counts.get("sun").unwrap();
let wind_count = *counts.get("wind").unwrap();
println!("{} {} {}", rain_count, sun_count, wind_count);
```

(A) 7 1 3

(B) 7 5 3

(C) 9 1 3

(D) 2 5 7

Question 38 [3 Marks]

What does this complete program print?

```
fn pair_total(a: &str, b: &str)
    -> Result<i32, std::num::ParseIntError>
{
    let first = a.parse::<i32>()?;
    let second = b.parse::<i32>()?;
    Ok(first + second)
}
fn main() {
    let first = pair_total("5", "7");
    let second = pair_total("bad", "9");
    let a = match first {
        Ok(n) => n,
        Err(_) => -1,
    };
    let b = match second {
        Ok(n) => n,
        Err(_) => -1,
    };
    println!("{} {}", a, b);
}
```

(A) It panics.

(B) 12 9

(C) 5 -1

(D) 12 -1

Question 39 [3 Marks]

What does this complete program print?

```
fn reading(text: &str) -> Result<i32, std::num::ParseIntError> {
    text.trim().parse()
}
fn fallback(text: &str, default: i32) -> i32 {
    match reading(text) {
        Ok(n) => n,
        Err(_) => default,
    }
}
fn main() {
    let mut last = 4;
    let mut total = 0;
    for text in ["6", "bad", " 2 "] {
        let current = fallback(text, last);
        total += current;
        last = current;
    }
    println!("{} {}", total, last);
}
```

(A) 14 2

(B) 14 4

(C) 12 2

(D) 8 2

Question 40 [3 Marks]

Which description of this complete program is correct? Ignore the exact panic message.

```
fn parse_count(text: &str) -> Result<u32, std::num::ParseIntError> {
    text.parse()
}
fn main() {
    let first = parse_count("4");
    match first {
        Ok(n) => println!("loaded {}", n),
        Err(_) => println!("invalid"),
    }
    let second = parse_count("bad");
    println!("checked");
    let count = second.expect("count required");
    println!("next {}", count);
    println!("done");
}
```

(A) It prints loaded 4 and checked, then panics.

(B) It prints loaded 4, then panics before checked.

(C) It prints loaded 4, checked, next 0, and done.

(D) It fails to compile because second contains an error.