Q1.1.

Please carefully read the following consent form and click "Next" to start the survey.

Consent for Exempt Research

The Pennsylvania State University

Title of Project: *An Investigation on Programmers' Rust Programming Experience and Challenges*
Principal Investigator: *Linhai Song*
Telephone Number: *814-863-7566*

You are being invited to participate in a research study. The study is voluntary. Its content and procedure are explained as follows.

- This research study is to understand programmers' Rust programming experience and reveal how it differentiates from the experience when writing programs in traditional programming languages (e.g., C/C++). In addition, the study also aims to investigate the challenges programmers face when they code Rust with the default development environment.
- Through the Qualtrics portal, you will be asked to visit a website constructed for this study. Several web pages will be presented to you. You will first fill in your demographics and evaluate your on-board experience in Rust. Then you will inspect several simple Rust programs, identify the embedded programming errors, and explain the reasons for the errors. In the end, you will answer a few post-session questions. You are expected to behave as you normally do during your daily Rust programming. Your responses will be collected by Qualtrics. Additionally, we will record some timing information such as how long it takes for each of your decisions using scripts embedded in the web pages.
- Efforts will be made to limit the use and the sharing of your personal research information to people who will review the information. No personally identifiable information will be collected during the study. Any personal information collected by the Qualtrics platform will not be associated with your answers. The general demographic data collected as part of this study will not be associated with your personal or identifiable information. In the event of any publication or presentation resulting from the research, no personally identifiable information will be shared. We will do our best to keep your participation in this research study confidential to the extent permitted by law. However, other people may notice your participation in this study, and some of the records could contain information that personally identifies you. Reasonable efforts will be made to keep your personal information private. However, absolute confidentiality cannot be guaranteed.

If you have questions, complaints, or concerns about the research, you can contact Linhai Song at 814-863-7566. If you have questions regarding your rights as a research subject or concerns regarding your privacy, you may contact the Office for Research Protections at 814-865-1775.

To participate in the study, you are required to be at least 18 years old, have previous Rust programming experience, and not currently be residing in the EEA. Your participation is voluntary, and you can decide to stop at any time. You do not have to answer any questions that you do not want to answer.

Your participation implies your voluntary consent to participate in the research.

Q296. Before you proceed to the survey, please complete the captcha below.

This question was not displayed to the respondent.

Q2.1.

Survey Procedure

This survey is to understand Rust programmers' on-board programming experience and the challenges they face.

This survey contains three phases. You will first fill in the demographic information and your experience in Rust programming for Phase 1. Then you will evaluate four small Rust programs for Phase 2. In the end, you will answer a few post-session questions for Phase 3. The survey will take about 20 minutes.

Q2.2.

Payment Instructions

At the end of the survey, you will be directed to another survey page. On the new survey page, you need to provide your contact information for the payment of a $10 Amazon Gift card.

Q3.1.

Phase 1.1: Demographic Information

Q3.2. 1.1 What is your age group?

- ○ Younger than 18 years old
- ○ 18 to 24 years old
- ○ 25 to 34 years old
- ○ 35 to 44 years old
- ○ 45 to 54 years old
- ○ 55 to 64 years old
- ○ 65 years and older
- ◉ Prefer not to answer

Q3.3. 1.2 What is your current location?

- ○ U.S.
- ○ European Economic Area (EEA)
- ○ China
- ○ Others
- ◉ Prefer not to answer

Q3.4. 1.3 What is your gender?

- ○ Male
- ○ Female
- ○ Non-binary / third gender
- ◉ Prefer not to answer

Q3.5. 1.4 How do you describe your ethnicity identity?

- ○ White
- ○ Asian
- ○ Black or African American
- ○ American Indian or Alaska Native
- ○ Hispanic or Latino
- ○ Native Hawaiian or Pacific Islander
- ○ Others
- ◉ Prefer not to answer

Q5.1.

Phase 1.2: On-board Experience of Rust

Q5.2.
1.5 How long have you learned to program Rust?

- ○ Have not learned at all
- ○ 0 - 6 months
- ○ 6 - 12 months
- ○ 1 - 2 years
- ○ 2 - 4 years
- ○ more than 4 years

- (●) Prefer not to answer

Q5.3. 1.6 How often do you work with Rust?

- ( ) Rarely
- ( ) Monthly
- ( ) Weekly
- ( ) Daily
- (●) Prefer not to answer

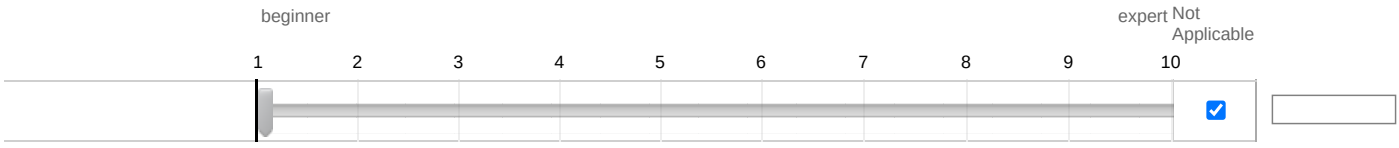
Q5.4. 1.7 How many lines of code are in the largest Rust program you have ever written?

- ( ) (0, 1000)
- ( ) (1000, 10000)
- ( ) ≥ 10000
- (●) Prefer not to answer

Q5.5. 1.8 Is Rust your most frequently used language now?

- ( ) Yes
- ( ) No
- (●) Prefer not to answer

Q5.6. 1.9 How do you rate your Rust expertise on a scale from 1 to 10? "1" means "beginner", and "10" means "expert".



Q5.7. 1.10 Do you find Rust's **lifetime** rules confusing when programming in Rust?

- ( ) Never
- ( ) Sometimes
- ( ) Most of the time
- ( ) Always
- (●) Prefer not to answer

Q5.8. 1.11 Do you find Rust's **ownership** rules confusing when programming in Rust?

- ( ) Never
- ( ) Sometimes
- ( ) Most of the time
- ( ) Always
- (●) Prefer not to answer

Q5.9. 1.12 Do you understand the error messages provided by the Rust compiler, when your code violates **lifetime** rules?

- ◯ Never
- ◯ Sometimes
- ◯ Most of the time
- ◯ Always
- ◉ Prefer not to answer

Q5.10. 1.13 Do you understand the error messages provided by the Rust compiler, when your code violates **ownership** rules?

- ◯ Never
- ◯ Sometimes
- ◯ Most of the time
- ◯ Always
- ◉ Prefer not to answer

Q5.11. 1.14 Besides lifetime and ownership, please write down other language features of Rust that confuse you when you program with Rust.

sadf

Q6.1.

Phase 2: Evaluating Rust Programs

You will evaluate four short Rust programs at Phase 2.
For the first two programs, you need to decide whether they can be compiled by the Rust compiler.
For the remaining two programs, you need to identify their embedded programming errors, evaluate the difficulty of identifying the errors and the helpfulness of the Rust compiler's error messages, and explain the violated safety rules.
This survey is NOT a test. We are interested in the difficulties and challenges that you have during Rust programming. We also hope to propose solutions to resolve those issues. When you answer the following questions, please DO NOT check or refer to any external resources.

Q6.2.

Phase 2.1: Identifying Rust Programs with Programming Errors

We will present two short Rust programs. Please decide whether they can be compiled by the Rust compiler.

Q7.1.
Rust Program 2:

```
fn main() {
    let my_array = vec![61, 14, 71, 23, 42, 8, 13, 66];
    let mx = max(my_array);
    let mn = min(my_array);
    println!("Max value is {}.", mx, mn);
}

fn max(array: Vec<i32>) -> i32 {
    71
}

fn min(array: Vec<i32>) -> i32 {
    8
}
```

Q7.2. Can the above program be compiled?

- ◯ Yes
- ◯ No
- ◉ Prefer not to answer

Q300.
Rust Program 2:

```
fn bar(x: &mut i32) {
    println!("{}", x);
}

fn main() {
    let mut a = 100;
    let y = &a;
    println!("{}", y);
    bar(&mut a);
}
```

Q301. Can the above program be compiled?

- ◯ Yes
- ◯ No
- ◉ Prefer not to answer

Q278.

Phase 2.2: Identifying Rust Programming Errors

Next, we will present two additional Rust programs. Both of them contain a programming error. For each program, you will

1. mark a statement or several tokens that you think cause the Rust compiler to reject the program;
2. specify the violated Rust safety mechanisms or safety rules;
3. evaluate the error messages reported by the Rust compiler and the difficulty level to identify the programming error;
4. and explain how the safety rules are violated.

This survey is NOT a test. We are interested in the difficulties and challenges that you have during Rust programming. We also hope to propose solutions to resolve those issues. When you answer the following questions, please DO NOT check or refer to any external resources.

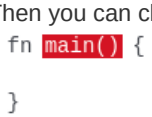
Q8.2.
You need to **highlight** tokens causing the programming error. Here is an example for demonstrating how to highlight a program token.

A simple Rust program is shown as below:

```
fn main() {

}
```

After you click "main()", a red button will pop up.

Then you can click the red button to mark “main()”



You can unmark a marked token by clicking the token and then clicking the “Remove” button.


```
fn main() {

}
```

Q279.
Ready for the second program?

Q9.1.
The following Rust program cannot be compiled.

a. Please mark the statement or the program tokens that cause the Rust compiler to reject the program.

This question was not displayed to the respondent.

Q9.2. b. Please rate how easy or how difficult it is to figure out the error of the above program on a scale from 1 to 10. "1" means "very easy", and “10" means "very difficult".

This question was not displayed to the respondent.

Q9.3. c. Which Rust safety rule is violated in the above program? "[X]" indicates a variable/type name in the program.
Only one of the following options is correct.

This question was not displayed to the respondent.

Q10.1. The following program is the same as the one on the previous page. We present the error messages reported by the Rust compiler afterward.

This question was not displayed to the respondent.

Q10.2.

```
#![allow(unused_variables)]

#[derive(Debug)]
struct Inner {
    in_a: u8,
    in_b: u8
}

struct Outer1 {
    a: [Inner; 2]
}

struct Outer2 {
    a: (Inner, Inner)
}

fn test(num: &mut u8, inner: &Inner) {
    *num += 1;
    println!("{:?}", inner);
}

fn main() {

    let mut out1 = Outer1 {
        a: [Inner {in_a: 1, in_b: 2}, Inner {in_a: 3, in_b: 4}]
    };

    let mut out2 = Outer2 {
        a: (Inner {in_a: 1, in_b: 2}, Inner {in_a: 3, in_b: 4})
    };

```

```
32      test(&mut out1.a[0].in_a, &out1.a[1]);
33      test(&mut out2.a.0.in_a, &out2.a.1);
34  }
35
```

This question was not displayed to the respondent.

Q10.3. Error messages:

```
error[E0502]: cannot borrow `out1.a[_]` as immutable because it is also borrowed as mutable
  --> main.rs:32:31
   |
32 |     test(&mut out1.a[0].in_a, &out1.a[1]);
   |     ---- ------------------  ^^^^^^^^^^ immutable borrow occurs here
   |     |    |
   |     |    mutable borrow occurs here
   |     mutable borrow later used by call
note: `a` is an array and can only be borrowed as a whole
   |
 9 | struct Outer1 {
10 |     a: [Inner; 2]
   |     ^

error: aborting due to previous error

For more information about this error, try `rustc --explain E0502`.
```

This question was not displayed to the respondent.

Q282. d. Given the above error messages, please rate how easy or how difficult it is to figure out the error of the above program on a scale from 1 to 10. "1" means "very easy", and “10" means "very difficult".

This question was not displayed to the respondent.

Q10.4. e. Please rate the helpfulness of the above error messages to your understanding of the error on a scale from 1 to 10. "1" means "not helpful", and "10" means "extremely helpful".

This question was not displayed to the respondent.

Q10.5. f. In your own words, please explain the above compiler error. Critically, please describe how the safety rule is violated, e.g., what program control constructs and data structures are involved, how the safety rule is applied to the context, and why the Rust compiler highlights some code.

This question was not displayed to the respondent.

Q11.1. The following program is the same as the one on the previous page. We present the error messages reported by the Rust compiler afterward.

This question was not displayed to the respondent.

Q11.2.

```
 1  #![allow(unused_variables)]
 2
 3  #[derive(Debug)]
 4  struct Inner {
 5      in_a: u8,
 6      in_b: u8
 7  }
 8
 9  struct Outer1 {
10      a: [Inner; 2]
11  }
12
13  struct Outer2 {
14      a: (Inner, Inner)
15  }
```

```
16
17  fn test(num: &mut u8, inner: &Inner) {
18      *num += 1;
19      println!("{:?}", inner);
20  }
21
22  fn main() {
23
24      let mut out1 = Outer1 {
25          a: [Inner {in_a: 1, in_b: 2}, Inner {in_a: 3, in_b: 4}]
26      };
27
28      let mut out2 = Outer2 {
29          a: (Inner {in_a: 1, in_b: 2}, Inner {in_a: 3, in_b: 4})
30      };
31
32      test(&mut out1.a[0].in_a, &out1.a[1]);
33      test(&mut out2.a.0.in_a, &out2.a.1);
34  }
35
```

This question was not displayed to the respondent.

Q11.3. Error messages:

```
error[E0502]: cannot borrow `out1.a[_]` as immutable because it is also borrowed as mutable
  --> main.rs:32:31
   |
32 |     test(&mut out1.a[0].in_a, &out1.a[1]);
   |     ---- ------------------  ^^^^^^^^^^ immutable borrow occurs here
   |     |    |
   |     |    mutable borrow occurs here
   |     mutable borrow later used by call

error: aborting due to previous error

For more information about this error, try `rustc --explain E0502`.
```

This question was not displayed to the respondent.

Q284. d. Given the above error messages, please rate how easy or how difficult it is to figure out the error of the above program on a scale from 1 to 10. "1" means "very easy", and “10" means "very difficult".

This question was not displayed to the respondent.

Q11.4. e. Please rate the helpfulness of the above error messages to your understanding of the error on a scale from 1 to 10. "1" means "not helpful", and "10" means "extremely helpful".

This question was not displayed to the respondent.

Q11.5. f. In your own words, please explain the above compiler error. Critically, please describe how the safety rule is violated, e.g., what program control constructs and data structures are involved, how the safety rule is applied to the context, and why the Rust compiler highlights some code.

This question was not displayed to the respondent.

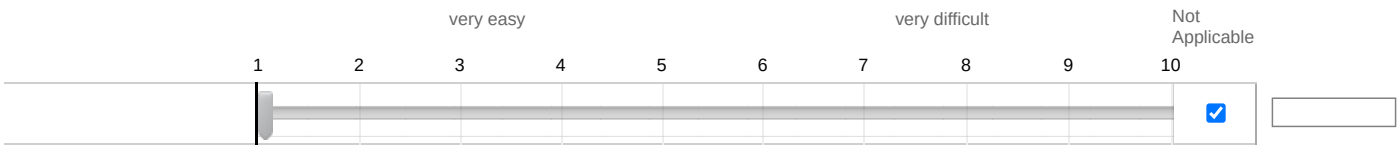
Q12.1.
The following Rust program cannot be compiled.

a. Please mark the statement or the program tokens that cause the Rust compiler to reject the program.

```
1  #![allow(unused_variables)]
2
3  #[derive(Debug)]
4  struct Inner {
5      in_a: u8,
```

```
 6     in_b: u8
 7 }
 8
 9 struct Outer1 {
10     a: [Inner; 2]
11 }
12
13 struct Outer2 {
14     a: (Inner, Inner)
15 }
16
17 fn test( inner: Inner, num: &mut u8) {
18     *num += 1;
19     println! ( "{:?}", inner);
20 }
21
22 fn main() {
23     let mut out1 = Outer1 {
24         a: [ Inner {in_a: 1, in_b: 2}, Inner { in_a: 3, in_b: 4 }]
25     };
26     let mut out2 = Outer2 {
27         a: ( Inner {in_a: 1, in_b: 2}, Inner { in_a: 3, in_b: 4 })
28     };
29
30     test( out1.a[1], & mut out1.a[0].in_a );
31     test( out2.a.1, & mut out2.a.0.in_a );
32 }
```

Q12.2. b. Please rate how easy or how difficult it is to figure out the error of the above program on a scale from 1 to 10. "1" means "very easy", and “10" means "very difficult".



Q12.3. c. Which Rust safety rule is violated in the above program? "[X]" indicates a variable/type name in the program.
Only one of the following options is correct.

- ○ two closures require unique access to [X] at the same time
- ○ cannot assign to [X] because it is borrowed
- ○ use of moved value: [X]
- ○ cannot move out of [X], a non-copy [X]
- ○ cannot return value referencing local variable [X]
- ○ closures cannot be static
- ○ field [X] specified more than once
- ○ cannot borrow [X] as immutable because it is also borrowed as mutable
- ○ lifetime of reference outlives lifetime of borrowed content
- ○ associated function [X] is private
- ◉ Prefer not to answer

Q13.1. The following program is the same as the one on the previous page. We present the error messages reported by the Rust compiler afterward.

Q13.2.

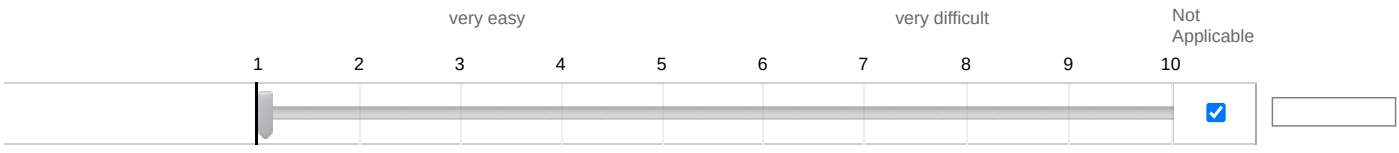
```
 1  #![allow(unused_variables)]
 2
 3  #[derive(Debug)]
 4  struct Inner {
```

```
 5      in_a: u8,
 6      in_b: u8
 7  }
 8
 9  struct Outer1 {
10      a: [Inner; 2]
11  }
12
13  struct Outer2 {
14      a: (Inner, Inner)
15  }
16
17  fn test(inner: Inner, num: &mut u8) {
18      *num += 1;
19      println!("{:?}", inner);
20  }
21
22  fn main() {
23      let mut out1 = Outer1 {
24          a: [Inner {in_a: 1, in_b: 2}, Inner {in_a: 3, in_b: 4}]
25      };
26      let mut out2 = Outer2 {
27          a: (Inner {in_a: 1, in_b: 2}, Inner {in_a: 3, in_b: 4})
28      };
29
30      test(out1.a[1], &mut out1.a[0].in_a);
31      test(out2.a.1, &mut out2.a.0.in_a);
32  }
```

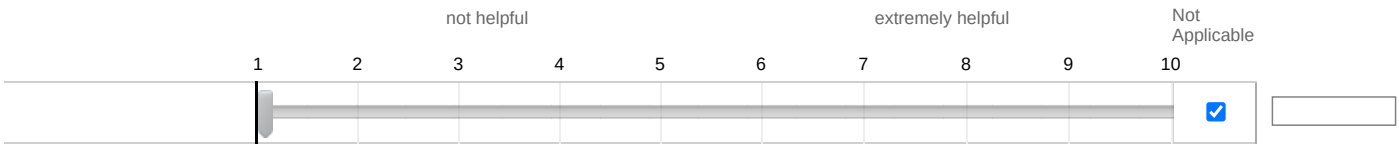
Q13.3. Error messages:

```
error[E0508]: cannot move out of type `[Inner; 2]`, a non-copy array
  --> main.rs:30:10
   |
30 |     test(out1.a[1], &mut out1.a[0].in_a);
   |          ^^^^^^^^^
   |          |
   |          cannot move out of here
   |          move occurs because `out1.a[_]` has type `Inner`, which does not implement the `Copy` trait

error: aborting due to previous error

For more information about this error, try `rustc --explain E0508`.
```

Q285. d. Given the above error messages, please rate how easy or how difficult it is to figure out the error of the above program on a scale from 1 to 10. "1" means "very easy", and “10" means "very difficult".



Q13.4. e. Please rate the helpfulness of the above error messages to your understanding of the error on a scale from 1 to 10. "1" means "not helpful", and "10" means "extremely helpful".



Q13.5. f. In your own words, please explain the above compiler error. Critically, please describe how the safety rule is violated, e.g., what program control constructs and data structures are involved, how the safety rule is applied to the context, and why the Rust compiler highlights some code.

fgh

Q14.1.
The following Rust program cannot be compiled.

a. Please mark the statement or the program tokens that cause the Rust compiler to reject the program.

This question was not displayed to the respondent.

Q14.2. b. Please rate how easy or how difficult it is to figure out the error of the above program on a scale from 1 to 10. "1" means "very easy", and “10" means "very difficult".

This question was not displayed to the respondent.

Q14.3. c. Which Rust safety rule is violated in the above program? "[X]" indicates a variable/type name in the program.
Only one of the following options is correct.

This question was not displayed to the respondent.

Q15.1. The following program is the same as the one on the previous page. We present the error messages reported by the Rust compiler afterward.

This question was not displayed to the respondent.

Q15.2.

```
 1  #![allow(unused_variables)]
 2
 3  #[derive(Debug)]
 4  struct Inner {
 5      in_a: u8,
 6      in_b: u8
 7  }
 8
 9  struct Outer1 {
10      a: [Inner; 2]
11  }
12
13  struct Outer2 {
14      a: (Inner, Inner)
15  }
16
17  fn main() {
18      let mut out1 = Outer1 {
19          a: [Inner {in_a: 1, in_b: 2}, Inner {in_a: 3, in_b: 4}]
20      };
21      let mut out2 = Outer2 {
22          a: (Inner {in_a: 1, in_b: 2}, Inner {in_a: 3, in_b: 4})
23      };
24      let r1 = &mut out1.a[0].in_a;
25      let r3 = &mut out2.a.0.in_a;
26      let r2 = &out1.a[1];
27      let r4 = &out2.a.1;
28      *r1 += 1;
29      *r3 += 1;
30
31      println!("{:?}", r2);
32      println!("{:?}", r4);
```

```
33  }
```

This question was not displayed to the respondent.

Q15.3. Error messages:

```
error[E0502]: cannot borrow `out1.a[_]` as immutable because it is also borrowed as mutable
  --> main.rs:26:14
   |
24 |     let r1 = &mut out1.a[0].in_a;
   |              ------------------- mutable borrow occurs here
25 |     let r3 = &mut out2.a.0.in_a;
26 |     let r2 = &out1.a[1];
   |              ^^^^^^^^^^ immutable borrow occurs here
27 |     let r4 = &out2.a.1;
28 |     *r1 += 1;
   |     -------- mutable borrow later used here

error: aborting due to previous error

For more information about this error, try `rustc --explain E0502`.
```

This question was not displayed to the respondent.

Q286. d. Given the above error messages, please rate how easy or how difficult it is to figure out the error of the above program on a scale from 1 to 10. "1" means "very easy", and “10" means "very difficult".

This question was not displayed to the respondent.

Q15.4. e. Please rate the helpfulness of the above error messages to your understanding of the error on a scale from 1 to 10. "1" means "not helpful", and "10" means "extremely helpful".

This question was not displayed to the respondent.

Q15.5. f. In your own words, please explain the above compiler error. Critically, please describe how the safety rule is violated, e.g., what program control constructs and data structures are involved, how the safety rule is applied to the context, and why the Rust compiler highlights some code.

This question was not displayed to the respondent.

Q16.1.
The following Rust program cannot be compiled.

a. Please mark the statement or the program tokens that cause the Rust compiler to reject the program.

This question was not displayed to the respondent.

Q16.2. b. Please rate how easy or how difficult it is to figure out the error of the above program on a scale from 1 to 10. "1" means "very easy", and “10" means "very difficult".

This question was not displayed to the respondent.

Q16.3. c. Which Rust safety rule is violated in the above program? "[X]" indicates a variable/type name in the program.
Only one of the following options is correct.

This question was not displayed to the respondent.

Q17.1. The following program is the same as the one on the previous page. We present the error messages reported by the Rust compiler afterward.

This question was not displayed to the respondent.

Q17.2.

```
#![allow(unused_variables)]

mod case2 {
    #[derive(Debug)]
    struct Foo {}

    struct Bar2<'b> {
        x: &'b Foo,
        i: i32
    }

    impl<'b> Bar2<'b> {
        fn f1(&'b mut self) -> &'b Foo {
            println!("{:?}", self.x);
            self.x
        }

        fn f2<'a>(&self, s: &'a str, ind: usize) -> &'a str {
            println!("{:?}, {}", self.x, s);
            &s[..ind]
        }

    }

    fn f4() {
        let foo = Foo {};
        let mut bar2 = Bar2 { x: &foo, i: 1 };
        let excerpt = bar2.f2("An online article", 2);
        println!("{}", excerpt);
        bar2.f1();
        let z = bar2.f1();
    }
}

fn main() {}
```

This question was not displayed to the respondent.

Q17.3. Error messages:

```
error[E0499]: cannot borrow `bar2` as mutable more than once at a time
  --> main.rs:31:17
   |
30 |         bar2.f1();
   |         ---- first mutable borrow occurs here
31 |         let z = bar2.f1();
   |                 ^^^^
   |                 |
   |                 second mutable borrow occurs here
   |                 first borrow later used here
note: the lifetime of reference `self` is the same as the referenced `Bar2` object
   |
12 |         impl<'b> Bar2<'b> {
   |                       -- note: `'b` is the lifetime parameter of struct `Bar2`
13 |             fn f1(&'b mut self) -> &'b Foo {
   |                    ^^
   |                    |
   |                    note: ...so this lifetime parameter makes the lifetime of
reference `self` the same as the referenced `Bar2` object

error: aborting due to previous error

For more information about this error, try `rustc --explain E0499`.
```

This question was not displayed to the respondent.

Q287. d. Given the above error messages, please rate how easy or how difficult it is to figure out the error of the above program on a scale from 1 to 10. "1" means "very easy", and “10" means "very difficult".

This question was not displayed to the respondent.

Q17.4. e. Please rate the helpfulness of the above error messages to your understanding of the error on a scale from 1 to 10. "1" means "not helpful", and "10" means "extremely helpful".

This question was not displayed to the respondent.

Q17.5. f. In your own words, please explain the above compiler error. Critically, please describe how the safety rule is violated, e.g., what program control constructs and data structures are involved, how the safety rule is applied to the context, and why the Rust compiler highlights some code.

This question was not displayed to the respondent.

Q18.1. The following program is the same as the one on the previous page. We present the error messages reported by the Rust compiler afterward.

This question was not displayed to the respondent.

Q18.2.

```
 1  #![allow(unused_variables)]
 2
 3  mod case2 {
 4      #[derive(Debug)]
 5      struct Foo {}
 6
 7      struct Bar2<'b> {
 8          x: &'b Foo,
 9          i: i32
10      }
11
12      impl<'b> Bar2<'b> {
13          fn f1(&'b mut self) -> &'b Foo {
14              println!("{:?}", self.x);
15              self.x
16          }
17
18          fn f2<'a>(&self, s: &'a str, ind: usize) -> &'a str {
19              println!("{:?}, {}", self.x, s);
20              &s[..ind]
21          }
22
23      }
24
25      fn f4() {
26          let foo = Foo {};
27          let mut bar2 = Bar2 { x: &foo, i: 1 };
28          let excerpt = bar2.f2("An online article", 2);
29          println!("{}", excerpt);
30          bar2.f1();
31          let z = bar2.f1();
32      }
33  }
34
35  fn main() {}
```

This question was not displayed to the respondent.

Q18.3. Error messages:

```
error[E0499]: cannot borrow `bar2` as mutable more than once at a time
  --> main.rs:31:17
   |
30 |         bar2.f1();
   |         ---- first mutable borrow occurs here
31 |         let z = bar2.f1();
   |                 ^^^^
   |                 |
   |                 second mutable borrow occurs here
   |                 first borrow later used here

error: aborting due to previous error
```

```
For more information about this error, try `rustc --explain E0499`.
```

This question was not displayed to the respondent.

Q288. d. Given the above error messages, please rate how easy or how difficult it is to figure out the error of the above program on a scale from 1 to 10. "1" means "very easy", and “10" means "very difficult".

This question was not displayed to the respondent.

Q18.4. e. Please rate the helpfulness of the above error messages to your understanding of the error on a scale from 1 to 10. "1" means "not helpful", and "10" means "extremely helpful".

This question was not displayed to the respondent.

Q18.5. f. In your own words, please explain the above compiler error. Critically, please describe how the safety rule is violated, e.g., what program control constructs and data structures are involved, how the safety rule is applied to the context, and why the Rust compiler highlights some code.

This question was not displayed to the respondent.

Q19.1.
The following Rust program cannot be compiled.

a. Please mark the statement or the program tokens that cause the Rust compiler to reject the program.

This question was not displayed to the respondent.

Q19.2. b. Please rate how easy or how difficult it is to figure out the error of the above program on a scale from 1 to 10. "1" means "very easy", and “10" means "very difficult".

This question was not displayed to the respondent.

Q19.3. c. Which Rust safety rule is violated in the above program? "[X]" indicates a variable/type name in the program.
Only one of the following options is correct.

This question was not displayed to the respondent.

Q20.1. The following program is the same as the one on the previous page. We present the error messages reported by the Rust compiler afterward.

This question was not displayed to the respondent.

Q20.2.

```
 1  #![allow(unused_variables)]
 2
 3  mod case2 {
 4      #[derive(Debug)]
 5      struct Foo {}
 6
 7      struct Bar2<'b> {
 8          x: &'b Foo,
 9          i: i32
10      }
11
12      impl<'b> Bar2<'b> {
13          fn f1(self) -> i32 {
14              println!("{:?}", self.x);
15              self.i
16          }
17
18          fn f2<'a>(&self, s: &'a str, ind: usize) -> &'a str {
19              println!("{:?}, {}", self.x, s);
20              &s[..ind]
```

```
21          }
22
23      }
24
25      fn f4() {
26          let foo = Foo {};
27          let bar2 = Bar2 { x: &foo, i: 1 };
28          let excerpt = bar2.f2("An online article", 2);
29          println!("{}", excerpt);
30          bar2.f1();
31          let z = bar2.f1();
32      }
33  }
34
35  fn main() {}
```

This question was not displayed to the respondent.

Q20.3. Error messages:

```
error[E0382]: use of moved value: `bar2`
  --> main.rs:31:17
   |
27 |          let bar2 = Bar2 { x: &foo, i: 1 };
   |              ---- move occurs because `bar2` has type `Bar2<'_>`, which does not implement the `Copy` trait
...
30 |          bar2.f1();
   |               ---- `bar2` moved due to this method call
31 |          let z = bar2.f1();
   |                   ^^^^ value used here after move
   |
note: this function consumes the receiver `self` by taking ownership of it, which moves `bar2`
  --> main.rs:13:15
   |
13 |          fn f1(self) -> i32 {
   |                ^^^^

error: aborting due to previous error

For more information about this error, try `rustc --explain E0382`.
```

This question was not displayed to the respondent.

Q289. d. Given the above error messages, please rate how easy or how difficult it is to figure out the error of the above program on a scale from 1 to 10. "1" means "very easy", and “10" means "very difficult".

This question was not displayed to the respondent.

Q20.4. e. Please rate the helpfulness of the above error messages to your understanding of the error on a scale from 1 to 10. "1" means "not helpful", and "10" means "extremely helpful".

This question was not displayed to the respondent.

Q20.5. f. In your own words, please explain the above compiler error. Critically, please describe how the safety rule is violated, e.g., what program control constructs and data structures are involved, how the safety rule is applied to the context, and why the Rust compiler highlights some code.

This question was not displayed to the respondent.

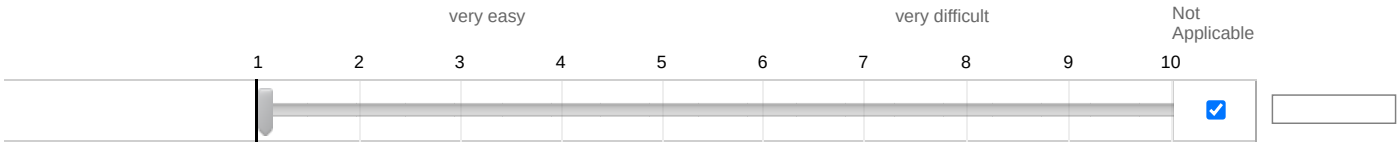
Q21.1.
The following Rust program cannot be compiled.

a. Please mark the statement or the program tokens that cause the Rust compiler to reject the program.

```
1  #![allow(unused_variables)]
2
3  mod case2 {
```

```
4   #[derive(Debug)]
5   struct Foo {}
6
7   struct Bar2<'b> {
8       x: &'b Foo,
9       i: i32
10  }
11
12  fn f4() {
13      let foo = Foo {};
14      let mut bar2 = Bar2 { x: &foo, i: 1 };
15      let f2: for<'a> fn( &Bar2, &'a str, usize) -> &'a str =
16          |bar: &Bar2, s: &str, ind: usize| {
17              println! ( "{:?}, {}", bar.x, s);
18              & s[..ind]
19          };
20      let f3: for<'a> fn( &'a mut Bar2 <'a> ) -> &'a Foo =
21          |bar: &mut Bar2| {
22              println!( "{:?}", bar.x);
23              bar.x
24          };
25      let excerpt = f2( &bar2, "An online article", 2);
26      println!( "{}", excerpt);
27      f3( &mut bar2);
28      let z = f3( &mut bar2);
29  }
30 }
31
32 fn main() {}
```

Q21.2. b. Please rate how easy or how difficult it is to figure out the error of the above program on a scale from 1 to 10. "1" means "very easy", and “10" means "very difficult".



Q21.3. c. Which Rust safety rule is violated in the above program? "[X]" indicates a variable/type name in the program.

Only one of the following options is correct.

- ○ closure may outlive the current function, but it borrows [X], which is owned by the current function
- ○ cannot use [X] because it was mutably borrowed
- ○ the trait `Copy` may not be implemented for this type
- ○ lifetime of reference outlives lifetime of borrowed content
- ○ multiple applicable items in scope
- ○ the parameter type [X] may not live long enough
- ○ cannot move out of [X] because it is borrowed
- ○ cannot move out of [X] which is behind a mutable reference
- ○ cannot borrow [X] as mutable more than once at a time
- ○ wrong number of type arguments
- ◉ Prefer not to answer

Q22.1. The following program is the same as the one on the previous page. We present the error messages reported by the Rust compiler afterward.

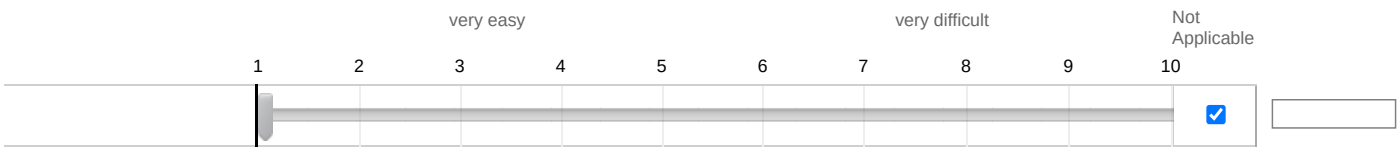
Q22.2.

```
 1  #![allow(unused_variables)]
 2
 3  mod case2 {
 4      #[derive(Debug)]
 5      struct Foo {}
 6
 7      struct Bar2<'b> {
 8          x: &'b Foo,
 9          i: i32
10      }
11
12      fn f4() {
13          let foo = Foo {};
14          let mut bar2 = Bar2 { x: &foo, i: 1 };
15          let f2: for<'a> fn(&Bar2, &'a str, usize) -> &'a str =
16              |bar: &Bar2, s: &str, ind: usize| {
17                  println!("{:?}, {}", bar.x, s);
18                  &s[..ind]
19              };
20          let f3: for<'a> fn(&'a mut Bar2<'a>) -> &'a Foo =
21              |bar: &mut Bar2| {
22                  println!("{:?}", bar.x);
23                  bar.x
24              };
25          let excerpt = f2(&bar2, "An online article", 2);
26          println!("{}", excerpt);
27          f3(&mut bar2);
28          let z = f3(&mut bar2);
29      }
30  }
31
32  fn main() {}
```

Q22.3. Error messages:

```
error[E0499]: cannot borrow `bar2` as mutable more than once at a time
  --> main.rs:28:20
   |
27 |          f3(&mut bar2);
   |             --------- first mutable borrow occurs here
28 |          let z = f3(&mut bar2);
   |                     ^^^^^^^^^
   |                     |
   |                     second mutable borrow occurs here
   |                     first borrow later used here

error: aborting due to previous error

For more information about this error, try `rustc --explain E0499`.
```

Q290. d. Given the above error messages, please rate how easy or how difficult it is to figure out the error of the above program on a scale from 1 to 10. "1" means "very easy", and “10" means "very difficult".



Q22.4. e. Please rate the helpfulness of the above error messages to your understanding of the error on a scale from 1 to 10. "1" means "not helpful", and "10" means "extremely helpful".

Q22.5. f. In your own words, please explain the above compiler error. Critically, please describe how the safety rule is violated, e.g., what program control constructs and data structures are involved, how the safety rule is applied to the context, and why the Rust compiler highlights some code.

as

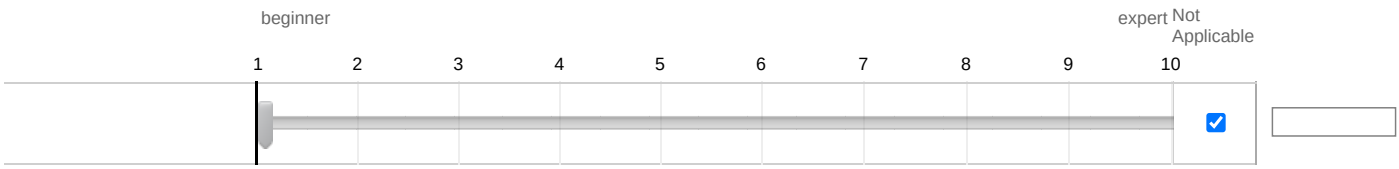
Q23.1.
Phase 3: Post-session Questions
In the end, please answer a few post-session questions about your programming experience in general.

Q295.
Post-session Questions

Q23.2. 3.1 How long have you learned to program?

- ○ 0 - 0.5 years
- ○ 0.5 years - 3 years
- ○ 3 years - 10 years
- ○ More than 10 years
- ◉ Prefer not to answer

Q23.3. 3.2 How do you rate your programming expertise on a scale from 1 to 10? "1" means "beginner", and "10" means "expert"?



Q23.4. 3.3 How many lines of code are in the largest program you have ever written?

- ○ (0, 1000)
- ○ (1000, 10000)
- ○ ≥ 10000
- ◉ Prefer not to answer

Q23.5. 3.4 Which programming language(s) do you have extensive experience in besides Rust? Check all that apply.

- ☐ C/C++
- ☐ Java
- ☐ R
- ☐ Python
- ☐ JavaScript
- ☐ Go

- [ ] C#
- [ ] Swift
- [ ] Haskell
- [ ] PHP
- [ ] Kotlin
- [ ] Ruby
- [ ] Objective-C
- [ ] Scala
- [ ] Perl
- [ ] Julia
- [ ] Others (please specify) _____
- [ ] None
- [x] Prefer not to answer

Q23.6.
3.5 Which title do you believe best matches your role?

- ( ) Programmer / Software Engineer
- ( ) Systems Architect
- ( ) DevOps / Site Reliability
- ( ) Manager / Team Leader
- ( ) Hobbyist
- ( ) Student
- ( ) Others (please specify) _____
- (•) Prefer not to answer

Q23.7. 3.6 Why do you choose to use/learn Rust? Check all that apply.

- [ ] Performance: Rust can offer performance comparable to C/C++
- [ ] Safety: memory safety, thread safety, and type safety
- [ ] Targeting bare-metal: Rust can be used to write an OS kernel or a device driver
- [ ] Compatibility: Rust is extremely backwards compatible and can be easily integrated with existing code in other languages
- [ ] Language Features: (im)mutability, traits, pattern matching, functional programming styles...
- [ ] Good dependency management and build tool
- [ ] Good documentation
- [ ] Others (please specify) _____
- [x] Prefer not to answer

Embedded Data

custom_slider_layout: 1

snippet_count: second

open_question_text: f. In your own words, please explain the above compiler error. Critically, please describe how the safety rule is violated, e.g., what program control constructs and data structures are involved, how the safety rule is applied to the context, and why the Rust compiler highlights some code.

difficulty_rating_text: b. Please rate how easy or how difficult it is to figure out the error of the above program on a scale from 1 to 10. "1" means "very easy", and “10" means "very difficult".

error_choosing_text: c. Which Rust safety rule is violated in the above program? "[X]" indicates a variable/type name in the program.
    Only one of the following options is correct.

second_difficulty_rating_text: d. Given the above error messages, please rate how easy or how difficult it is to figure out the error of the above program on a scale from 1 to 10. "1" means "very easy", and “10" means "very difficult".

simple_snippet_count: 2

helpful_text: e. Please rate the helpfulness of the above error messages to your understanding of the error on a scale from 1 to 10. "1" means "not helpful", and "10" means "extremely helpful".

Referer: https://pennstate.ca1.qualtrics.com/jfe/previewForm/SV_3vFE9UIfDmLIS4C?Q_CHL=preview&Q_SurveyVersionID=current

Location Data

Location: (40.795700073242, -77.861801147461)

Source: GeoIP Estimation