A SURVEY INSTRUMENTS

A.1 (Pre-survey Introduction)

A.1.1 Please carefully read the following consent form and click "Next" to start the survey.

[Show The consent form]

A.1.2 Before you proceed to the survey, please complete the captcha below.

[Show Captcha]

A.1.3 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.

A.1.4 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.

A.2 Phase 1.1: Demographic Information

A.2.1 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

A.2.2 1.2 What is your current location?

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

A.2.3 1.3 What is your gender?

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

A.2.4 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

A.3 On-board Experience of Rust

A.3.1 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

A.3.2 1.6 How often do you work with Rust?

○ Rarely ○ Monthly ○ Weekly ○ Daily
○ Prefer not to answer

A.3.3 1.7 How many lines of code are in the largest Rust program you have ever written?

○ (0, 1000) ○ (1000, 10000) ○ $\geq$ 10000
○ Prefer not to answer

A.3.4 1.8 Is Rust your most frequently used language now?

○ Yes ○ No ○ Prefer not to answer

A.3.5 1.9 How do you rate your Rust expertise on a scale from 1 to 10? "1" means "beginner", and "10" means "expert".

[10-point scale slider with a "Prefer not to answer" check box]

A.3.6 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

A.3.7 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

A.3.8 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

A.3.9 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

A.3.10 1.14 Besides lifetime and ownership, please write down other language features of Rust that confuse you when you program with Rust.

[Text Box]

A.4 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.

A.5 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.

A.5.1 Rust Program 1[or 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
}
```

[PA is shown here.]

A.5.2 Can the above program be compiled?

○ Yes ○ No ○ Prefer not to answer

A.6 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.

[Page Break]

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()".

```
fn main() {

}
```

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

A.7 Ready for the first[or second] program?

A.8 The following Rust program cannot be compiled.

A.8.1 a. Please mark the statement or the program tokens that cause the Rust compiler to reject the program.

```
#![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})
    };

    test(&mut out1.a[0].in_a, &out1.a[1]);
    test(&mut out2.a.0.in_a, &out2.a.1);
}

```

[PC-1 is shown here. Participants could highlight tokens on the program.]

A.8.2 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".

[10-point scale slider with a "Prefer not to answer" check box]

A.8.3 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.

[Choices are shown in a randomized order, with an additional "Prefer not to answer" always shown in the end.]

- ○ cannot borrow [X] as immutable because it is also borrowed as mutable
- ○ cannot move out of [X], a non-copy [X]
- ○ cannot assign to [X] because it is borrowed
- ○ cannot return value referencing local variable [X]

- ○ lifetime of reference outlives lifetime of borrowed content
- ○ use of moved value: [X]
- ○ two closures require unique access to [X] at the same time
- ○ field [X] specified more than once
- ○ associated function [X] is private
- ○ closures cannot be static

[The page breaks here so that the following three questions are shown in the next page.]

A.8.4 The following program is the same as the one on the previous page. We present the error messages reported by the Rust compiler afterward.

[Show the program and error message. If PC-1 and PD-1 is showing, one of the enhanced error message or the original error message is randomly shown here. Otherwise, the default error message is shown here.]

A.8.5 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".

[10-point scale slider with a "Prefer not to answer" check box]

A.8.6 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".

[10-point scale slider with a "Prefer not to answer" check box]

A.8.7 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.

[Text box]

A.9 Phase 3: Post-session Questions

In the end, please answer a few post-session questions about your programming experience in general.

A.10 Post-session Questions

A.10.1 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

A.10.2 3.2 How do you rate your programming expertise on a scale from 1 to 10? "1" means "beginner", and "10" means "expert"?

[10-point scale slider with a "Prefer not to answer" check box]

A.10.3 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

A.10.4 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) [Text box] □ None
□ Prefer not to answer

A.10.5 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) [Text box] ○ Prefer not to answer

A.10.6 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) [Text box]
- □ Prefer not to answer