Please note that this survey requires at least a 13" monitor and cannot be completed on a mobile phone.

Consent Form

By completing this online questionnaire (i.e., clicking “Next”), you agree to participate in the study.

The study focuses on investigating the type and amount of information needed for debugging program failures. The study is structured in the form of an online questionnaire with 12 questions, which are split into three parts. Each part includes questions about the same piece of code. The entire survey is expected to take around 20-30 minutes of your time.

As a “thank you”, we will randomly select 10 participants who finished the study to receive a **$30 Amazon gift card**. Moreover, your participation in this study will help the academic and industrial community gain valuable insight into debugging approaches. The results will be fully anonymized and will be reported in an academic paper, which will be openly available to the community.

Participation is completely voluntary and you may withdraw from the study at any time before the final reports are made public.

We do not collect identifying information in this study. However, **if you would like to be considered for the award, withdraw your data from the study at a later stage, and/or be notified when the results of the study are available, you should provide your email address as contact information**. We will also appreciate it if you provide optional demographic information, which will only be shared in an aggregated form and will not be associated with any individual responses.

Thank you in advance for your time.

Your email address (optional):

Experience and Demographics

1. Which of the following describes you the best? (pick all that apply)

- ☐ Software developer or engineer working in industry
- ☐ Software tester working in industry
- ☐ Researcher working in industry (Research Staff Member, Research Fellow, Research Engineer)
- ☐ Researcher working in academia, non-student (Postdoctoral Fellow, Faculty Member)
- ☐ PhD student
- ☐ Master's student
- ☐ Undergraduate student
- ☐ Other

2. How many years of programming experience while **in school/university** do you have? (in any programming language)

- ○ No experience
- ○ Less than 1 year
- ○ At least 1 but less than 3 years
- ○ At least 3 but less than 5 years
- ○ At least 5 but less than 10 years
- ○ More than 10 years

3. How many years of programming experience **outside of school/university** do you have?

- ○ No experience
- ○ Less than 1 year
- ○ At least 1 but less than 3 years
- ○ At least 3 but less than 5 years
- ○ At least 5 but less than 10 years
- ○ More than 10 years

4. How would you rate your programming skill level?

- ○ Novice: developed a few small programs
- ○ Intermediate: developed a few large programs
- ○ Advanced: developed several large software systems

5. What is your software development area? (e.g., web developer, full stack developer, embedded systems developer, ML data analyst)

6. How do you debug your code? (Pick all that apply)

- ☐ I do not debug my code
- ☐ Program logging (e.g., print)
- ☐ Assertions
- ☐ IDE debugger utilities (e.g., breakpoints and stepping)
- ☐ Other

7. What is your country of employment or studies?

8. What is your age?

- ○ <25
- ○ 25-34
- ○ 35-44
- ○ 45-54

- 55-64
- >64
- Prefer not to answer

9. What is your gender?

- Female
- Male
- Non-binary person
- Prefer not to answer

Part I

Part 1/3: Explaining the failure

In this part of the survey, you are given two code snippets: the old version of the code (V1), where an assertion passes, and a new version (V2), where the same assertion fails due to changes made in the code. The goal of this study is to help developers understand **why** changes made in this code resulted in assertion failure.

1. Please select statements you deem important for understanding and debugging the failure (in addition to the changed statements that are clearly important and, thus, already pre-selected below). Click on a statement to select it and click again to deselect it.

Notations: Lines that correspond to each other in V1 and V2 are given the same line numbers. Numbered empty lines indicate statements that were deleted in a version. Changes between versions ("Add", "Update", "Delete") are shown on arrows between code snippets. For simplicity, each "*if*" statement (e.g., in line 9) is annotated with a label showing whether the "*if*" condition is evaluated to *true* or *false*. Gray lines indicate statements that are not executed because they are encapsulated by an "*if*" statement that evaluates to *false*.

V1

```
1  public static void main(String[] args){
2      int real = POS_INFINITY;
3      int img = 1;
4      assertSame(new Complex(1, 1), tanh(real, img));
5  }
6  public Complex tanh(int real, int img){
7      int real2 = 0;
8      int img2 = 0;
9      if (real <= 20){ [false]
10         real2 = exp(real)/2;
11         int k = (int)(img * 0.6366197725814);
12         int a = -k * 1.570796251296997;
13         int remA = img + a;
14         a = -k * 7.549789948768648E-8;
15         int b = remA;
16         remA = a + b;
17         int remB = -(remA*2);
18         img2 = remB *5;
19     }
20     if (img >= 1){ [true]
21         real2 = real2 + img;
22         img2 = img2 + img;
23     }
24     return new Complex(real2,img2);
25 }
```

Update

Delete

V2

```
1  public static void main(String[] args){
2      int real = POS_INFINITY;
3      int img = 1;
4      assertSame(new Complex(1, 1), tanh(real, img));
5  }
6  public Complex tanh(int real, int img){
7      int real2 = 0;
8      int img2 = 0;
9      if (real >= 20){ [true]
10         real2 = exp(real)/2;
11         int k = (int)(img * 0.6366197725814);
12         int a = -k * 1.570796251296997;
13         int remA = img + a;
14         a = -k * 7.549789948768648E-8;
15         int b = remA;
16         remA = a + b;
17         int remB = -(remA*2);
18         img2 = remB *5;
19     }
20
21
22
23
24     return new Complex(real2,img2);
25 }
```

2. Please explain the failure in your own words. Specifically, please describe why changes made in this code resulted in the assertion failure.

3. Please explain why you selected these statements as relevant for understanding the failure.

Part II

Part 2/3: Comparing code views

In this part of the survey, you will be given three different views, named A, B, and C. Each view contains two versions of a code snippet: V1 and V2.

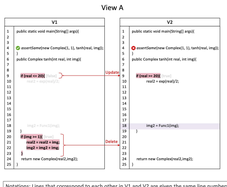
Unlike the code given in Part 1, each snippet now only includes a subset of code statements deemed relevant to the failure. Views A, B, and C differ by the subset of statements their code snippet includes. You will be asked to rank the view based on their:

- **Completeness**: include all essential information needed to explain and debug the failure.
- **Conciseness**: do not include unnecessary information, unneeded to explain and debug the failure.

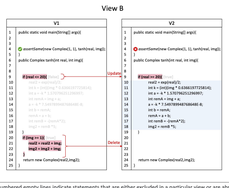
The goal is to identify views that are most helpful to explain and debug the failure.

Note: Clicking on the picture below opens its larger version.

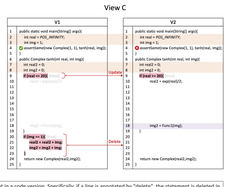
View A



View B



View C



Notations: Lines that correspond to each other in V1 and V2 are given the same line numbers. Numbered empty lines indicate statements that are either excluded in a particular view or are absent in a code version. Specifically, if a line is annotated by "delete", the statement is deleted in a version. Otherwise, it is excluded from the view. Changes between versions ("Add", "Update", "Delete") are shown on arrows between code snippets. For simplicity, each "if" statement (e.g., in line 9) is annotated with a label showing whether the "if" condition is evaluated to *true* or *false*. Gray lines indicate statements that are not executed because they are encapsulated by an "if" statement that evaluates to *false*. Colored backgrounds highlight the differences between the views.

4. Please rank views A, B, and C (1 being the best; 3 being the worst). You can drag the view names into the box

and then rank them internally. When ranking views, please consider:

- **Completeness**: include all essential information needed to explain and debug the failure.
- **Conciseness**: do not include unnecessary information, unneeded to explain and debug the failure.

Items

View A

View B

View C

Ranking Views

5. Explain your ranking by describing the advantages and disadvantages of each view (given in their sequential order below).

View A:
+
+
-
-
View B:
+
+
-
-
View C:
+
+
-
-

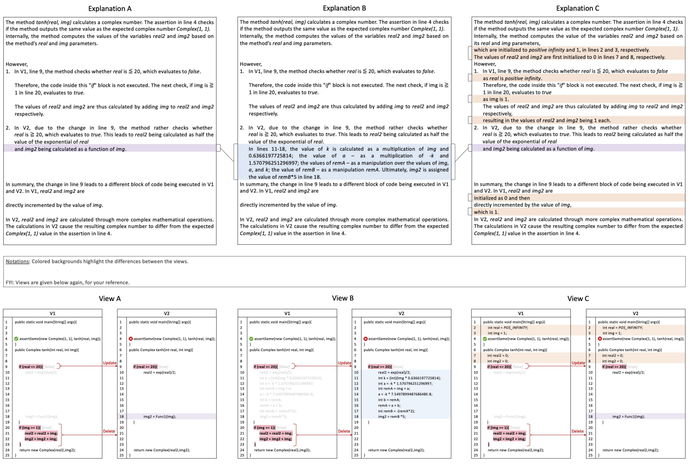
Part III

Part 3/3: Comparing textual explanations

In this part of the survey, you are given three different **textual** explanations of the failure. You will need to rank these explanations based on their:

- **Completeness**: include all essential information needed to explain and debug the failure.
- **Conciseness**: do not include unnecessary information, unneeded to explain and debug the failure.

Note: Clicking on the picture below opens its larger version.



6. Please focus on the **textual** explanation now (upper part of the picture). Please rank explanations A, B, and C (1 being the best; 3 being the worst). You can drag the explanation names into the box and then rank them internally.

Items

Explanation A

Explanation B

Explanation C

Ranking Explanations

7. Explain your ranking by describing the advantages and disadvantages of each explanation (given in their sequential order below).

Explanation A:
+
+
-
-
Explanation B:
+
+
-
-
Explanation C:
+
+
-
-

8. Which of the following you prefer to see when understanding and debugging the failure?

- ○ Code views
- ○ Textual explanations
- ○ Both

9. Explain your selection.

10. Have looking at the code views and/or reading the explanations changed your understanding of the failure and why? Would you now augment the explanation you gave in Part 1, Question 3 (you can navigate to your explanation by pressing the back button twice)

11. Please list any suggestions for how the views and textual explanations you liked the most can be improved even further.

12. Do you have any other comments related to this survey?