



Girls Who Code At Home

Debug the Missing Code Part 2
Reference Guide

Debug the Missing Code Part 2 Reference Guide



In this document you will find all of the answers to some of the questions in the activity. Follow along with the activity and when you see this icon, stop and check your ideas here.

Step 1: Meet the logic bugs

Why do logic errors cause problems?

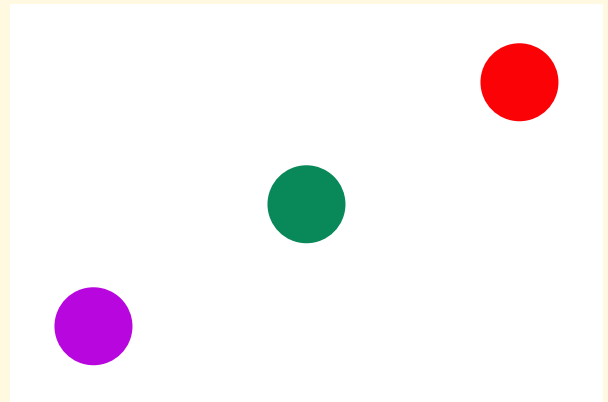
The color will be brownish because the program doesn't tell the painter to clean their brush after each use. It only tells them to clean the brush at the end after they are done painting.

For the last circle to be that vibrant purple, you would add a **Rinse brush** instruction after each color:

CODE

```
Dip brush in red paint
Paint circle
Rinse brush
Dip your brush in green paint
Paint circle
Rinse brush
Dip your brush in purple paint
Paint circle
Rinse brush
Let canvas dry
```

RESULTS



Step 4: Review the code

Below is a detailed explanation of the [broken code](#) at the beginning of the step.

Declare variables

First, we declared the variables to store the score for each bug (e.g. **butterflyScore**). Next, we declared a series of variables to store the HTML element for each of the answer buttons in our questions (e.g. **q2a3**). We also declared variables to store how many questions a person has clicked on (**questionCount**) and to store the result and restart HTML elements.

Listen for button clicks

Next, we added event listeners to all of our buttons. [Event listeners](#) listen for certain input on a webpage, like a mouse click, a keypress, or scrolling. If the browser “hears” or detects the input that it’s waiting for, the program takes an action. We define this action in a function that goes inside the event listener. For example, if a person chooses **Mochi** for Question 1, they click the **q1a1** button. When they click the button, the browser executes the code in the **butterfly** function.

JAVASCRIPT	DESCRIPTION
<code>q1a1.addEventListener("click", butterfly);</code>	<ul style="list-style-type: none">→ q1a1: Variable that stores input from the mochi button (the first answer to the first question).→ .: The period or dot tells our program to attach the variable to the method after it. Here that method is <code>addEventListener</code>.→ addEventListener: The keyword to call the event listener method (or function).→ "click": The event we are listening for.→ butterfly: The function that will execute if the event occurs.→ ;: All JavaScript statements end with a semicolon.

Track and update the score

Now we need to create the functions that we listed inside our event listeners. Four of these track the score of each bug: **bee**, **butterfly**, **grasshopper**, and **ladybug**. If one of these functions is called (i.e. when a person clicks a button with that function attached to it), we increase the bug’s score by 1 and we increase the **questionCount** variable by 1.

Next, we have a conditional statement to evaluate when the quiz is complete. If `questionCount` is equal to 3 (we use the [comparison operator](#) `==` here, not the assignment operator `=`), then we execute the **updateResult** function below. This function checks each score to see if it is greater than or equal to 2. If one of them meets that condition, this function updates the text at the bottom with your result. If none of them do, then we ask the person to try again.

Restart the quiz

The final function executes if the *Restart* button is clicked. Here, restart means resetting all the scores and the **questionCount** to 0. Otherwise, the numbers would continue increasing.

Step 5: Describe the problem

Below is an example of one way to describe what we want to happen in our program.

- When a person clicks an answer button, the program calls the specified bug function (i.e. **bee**, **butterfly**, **grasshopper**, or **ladybug**) attached to it in the event listener.
- The function adds 1 to the current score of that bug and adds 1 to the current **questionCount** value.
- Test If the **questionCount** value is equal to 3 each time the **questionCount** value increases.
Note: Since the quiz has three questions, we know a person has finished if this value equals 3. If we had four questions we would change it to 4.
- If **questionCount** is not equal to 3, don't do anything and allow the person to continue the quiz.
- If **questionCount** is equal to 3, call the **updateResult** function to test the value of each bug's score.
- Evaluate each bug's score and return a result in the **updateResult** function:
 - ◆ If **beeScore** is greater than or equal to 2, update the bottom text to say "You are a bee!"
 - ◆ Else if **butterflyScore** is greater than or equal to 2, update the bottom text to say "You are a butterfly!"
 - ◆ Else if **grasshopperScore** is greater than or equal to 2, update the bottom text to say "You are a grasshopper!"
 - ◆ Else if **ladybugScore** is greater than or equal to 2, update the bottom text to say "You are a ladybug!"
 - ◆ Else update the bottom text to say "Hmm...not sure. Try again later"
- When the *Restart* button is clicked, call the **restartQuiz** function set all variables to 0.

Step 6: Make it observable

Code with **console.log()** added to each scoring function:

```
58 // Track bee score
59 function bee() {
60     beeScore += 1;
61     questionCount += 1;
62     console.log("questionCount = " + questionCount + "\t" + "beeScore = " +
63     beeScore);
64 }
```

```

65 // Track butterfly score
66 function butterfly() {
67     butterflyScore += 1;
68     questionCount += 1;
69     console.log("questionCount = " + questionCount + "\t" + "butterflyScore = "
70     + butterflyScore);
71 }
72 // Track grasshopper score
73 function grasshopper() {
74     grasshopperScore += 1;
75     questionCount += 1;
76     console.log("questionCount = " + questionCount + "\t" + "grasshopperScore = "
77     + grasshopperScore);
78 }
79 // Track ladybug score
80 function ladybug() {
81     ladybugScore += 1;
82     questionCount += 1;
83     console.log("questionCount = " + questionCount + "\t" + "ladybugScore = "
84     + ladybugScore);
85 }

```

Answers to our clues:

- ☐ **What controls the result text?**
 - ☐ The **updateResult** function.
- ☐ **When do we call (or use) the updateResult function?**
 - ☐ In the conditional statement that evaluates when **questionCount** is equal to 3.
- ☐ **Is the conditional statement running when we need it to?**
 - ☐ Nope!

Step 8: Test one thing at a time

Hypothesis #1: Testing questionCount conditional in the bee function

```
58 // Track bee score
59 function bee() {
60     beeScore += 1;
61     questionCount += 1;
62     console.log("questionCount = " + questionCount + "\t" + "beeScore = " +
        beeScore);
63     // Add conditional
64     if (questionCount == 3){
65         updateResult();
66     }
67 }
```

Hypothesis #2: Testing questionCount conditional in remaining functions

```
69 // Track butterfly score
70 function butterfly() {
71     butterflyScore += 1;
72     questionCount += 1;
73     console.log("questionCount = " + questionCount + "\t" + "butterflyScore = "
        + butterflyScore);
74     // Add conditional
75     if (questionCount == 3){
76         updateResult();
77     }
78 }
79
80 // Track grasshopper score
81 function grasshopper() {
82     grasshopperScore += 1;
83     questionCount += 1;
84     console.log("questionCount = " + questionCount + "\t" + "grasshopperScore = "
        + grasshopperScore);
85     // Add conditional
86     if (questionCount == 3){
87         updateResult();
88     }
89 }
```

```
91 // Track ladybug score
92 function ladybug() {
93     ladybugScore += 1;
94     questionCount += 1;
95     console.log("questionCount = " + questionCount + "\t" + "ladybugScore = "
96               + ladybugScore);
97     // Add conditional
98     if (questionCount == 3){
99         updateResult();
100     }
101 }
```



Bug Fixed!

In this error, the program never evaluated the conditional that called the **updateResult** function. While it may seem logical to test the value of **questionCount** after all of the bug score functions, we actually need to test it each time a bug score function runs. We solved the problem by placing the conditional inside each bug score function so the program can check the value of **questionCount** after a person clicks a button.