# **Debugging Discussion Guide (for use during or after reading)**

1. When it comes to programming, what are bugs? Why is it important to learn how to fix them? (Pesky Bugs!, p. 4-5)
   1. In programming, a bug does not refer to a small insect, rather it refers to an error in a computer program. It is important for all programs to learn how to fix bugs because mistakes and errors happen! Because there are so many rules involved in writing code and so many lines of code to type, it can be easy for even the best programmers to make mistakes. Knowing how to find and fix bugs allows programmers to finalize programs without errors.
2. Why can it be difficult for programmers to find bugs? (Finding Bugs, p. 6-9)
   1. Finding a bug can be difficult because code can be long and complex. In order to find a bug, programmers must read through every line of code carefully to spot the error. Because code is written in high- and low-level languages, rather than human languages, the process of finding a bug can be even more challenging.
3. How do programmers find bugs? Describe the process of stepping using a debugger. (Finding Bugs, p. 6-9)
   1. Programmers find bugs by using a debugger and a process called stepping. A debugger is a special computer program that runs through a program’s code line by line while that program is running. Programmers can see each line of code as the debugger steps through them one by one. This allows them to see when and in which line of code an error occurs, allowing them to more easily identify and fix errors.
4. Provide an example of an easy or simple bug to fix as well as a more complex one. (Fixing Bugs, p. 10-11)
   1. Once programmers have found a bug, they must fix it. Some bugs are easy to fix. For example, sometimes code is missing a symbol or has instructions written in the wrong order. Programmers can fix these bugs by simply inserting the correct code. Other times, bugs can be more challenging to fix and might require a programmer to re-write the code so it better matches the program’s intended purposes.
5. What is syntax? When can syntax errors occur? (Syntax Errors, p. 12-13)
   1. Syntax in coding is similar to syntax in any language. In English, syntax refers to the order and arrangement of words in a sentence. In code, syntax refers to the order in each line of code as well as the order of the lines themselves. Syntax helps programmers create codes that computers can understand. Syntax errors can occur when there is not proper syntax. For example, if a code is written in the wrong order, one might get a syntax error warning. In addition, if a code uses the wrong word or symbol, one might also receive a syntax error warning. Coding can be a complicated and detailed process where mistakes that cause syntax errors are bound to happen, even to the world’s best programmers!
6. What is a token error? When might token errors occur? (Syntax Errors, p. 14-15)
   1. A token error is a specific type of syntax error. A token error is a type of bug caused when the code includes words or symbols that are not in the programming language. For example, in a series of steps for planting, writing the word “boots” instead of “roots” would be considered a token error because the program language does not include the word “boots.”
7. What is a logic error? What effect do logic errors typically have on code? (Logic Errors, p. 22-23)
   1. A logic error is a bug that does not stop a program from working, but does cause it to do the wrong thing. Logic errors occur when the steps of a program are in an incorrect order. Although each individual step may not have any errors, the code will likely not run correctly or produce the wanted outcome because the steps do not follow a logical order.
8. What are variables? What errors can occur if the variables in a program are not correct? (Logic Errors, p. 24-25)
   1. A variable is a value, or a piece of information, that can change. If the variables in a program are not correct, a logic error can occur. Instructions that include the wrong variable can often still be read and followed by the computer, but they may produce an outcome that is not ideal.
9. What advice does the text provide for preventing bugs? (Preventing Bugs, p. 26-29)
   1. One of the best ways to keep a program bug-free is prevention! In order to prevent bugs from happening, the text suggests programmers take their coding process slow, as it is easier to make mistakes when we rush. In addition, programmers can prevent bugs by keeping their codes as simple and straightforward as possible.
10. If you do find bugs in your programs, what can you tell yourself to support your growth mindset? (Preventing Bugs, p. 26-29)
    1. Students’ answers will vary.
    2. Even the best programmers find bugs in their programs! Mistakes happen! If you do find a mistake, it’s important to stay positive and support your growth mindset by reminding yourself that mistakes are opportunities to learn.