1. “in EVAL with 5.0 and 4”  
    The program first pushes the System.out.println(te.eval(“5.0”, “4”)); method call and then prints the first line from that method.
2. “in first catch, nfe”  
   The program then tried to take n1 (5.0) and convert it into an integer. However, because n1 is a floating-point number, it stepped to the NumberFormatException catch block and returned its error message.
3. “about to return, num = 1”  
   As part of the catch block, the num variable increments by 1, so when the program steps out of the catch block it prints the updated value of 1.
4. “B”  
   The program then returns the value array’s num position, 1, which is assigned to “B”.
5. “in EVAL with 5 and 0”  
   The program then pushes the String variables of “5” and “0”. It tries to convert both to integers (successfully) but then divides by zero, an illegal action.
6. “in second catch, ae”  
   Because dividing by zero throws an ArithmeticException, that catch block executes, printing the referenced line and incrementing the num variable a second time.
7. “about to return, num = 2”  
   The exception incremented the num variable again, making its new value 2.
8. “C”  
   The returned value array position is now index 2, or “C,” which is printed by that line.
9. “in EVAL with 22 and 5”  
   This time, when the JVM passes these numbers, there is no thrown exception. It steps into the try block, converts the strings to integers, and performs integer division to assign the value of 4 to the num variable.
10. “about to return, num = 4”  
    Again, because no exception was thrown, division was successful, and the num variable is assigned to 4.
11. “E”  
    The value array’s fourth index position of E is returned, which is then printed.
12. “in EVAL with 33 and 5”  
    Like 22 and 5, both of these numbers are integers and can be processed in the try method, which assigns num to 6.
13. “about to return, num = 6”  
    With num at 6, the eval method returns value array’s sixth index position. Because this is an IndexOutOfBounds exception, it moves to the catch block in the main method.
14. “in main’s catch”  
    Because there is an exception object, we step out of the try block and into main’s catch block, which prints this message.
15. “bye”  
    The program steps out of the catch block and executes the remaining code, which is a printing method.

After moving System.out.println(te.eval(“33”, “5”)); to the first line of main’s try block, we get four lines of code, which are the same as the last four lines of code in the original run of this program. This time, the system steps into the eval method, prints the line “in EVAL with 33 and 5”, and steps into the eval method’s try block. Since it can successfully convert the two numbers to integers and divide them, the num variable is assigned 6. The program then announces that it is going to return the information at value array’s sixth position. However, since this throws an IndexOutOfBoundsException, the exception object is sent back to main’s generic catch method, prints its message, and jumps out of the try-catch mechanism to execute the remaining code. It will not run the other eval methods because it has thrown an exception in the main method.