Sometimes what you think your program does and what it *actually* does are two different things. This worksheet is about some different ways of discovering what your code actually does.

Desk checking is the process of simulating the steps your code does without using a computer. You mentally execute each program statement, keeping track of the current value of each variable.

1) Desk check the following code. Begin by writing "x:", "y:", and "tmp:", each on its own line. Then pretend you are the computer and execute each line of code until the program is finished. Each time a variable is assigned, cross out the old value and write the new one on the variable's line.

```
int x = 12;
int y = 9;
while (y != 0) {
    int tmp = x % y;       // the remainder of x / y
    x = y;
    y = tmp;
}
System.out.println("GCD = " + x);
```

2) Desk check the following code

```
for (int i = 1; i <= 3; i++) {
    for (int j = 1; j <= 4; j++) {
        System.out.print((i * j) + " ");
    }
    System.out.println();
}
```

3) Desk check the following code.

```
int total = 25;
for (int number = 1; number <= (total / 2); number++) {
    total = total - number;
    System.out.println(total + " " + number);
}
```

Tracing using printing is not as illuminating but is faster than desk checking because it uses a computer. In this technique, you insert a line of code that prints your variables at some place strategic so that you can see their values as they change.

4) Verify your desk check answer in Problem 1 by inserting a println statement somewhere in the code and running it.

Good practice: Using your brain to desk check your code and then verifying that your code does what you think it does with your computer is a powerful way to ensure your intentions and the computer's interpretation are the same.

Tracing with a debugger is another way to see what your code is doing. Most integrated development environments (IDEs) have a built-in debugger that allows you to stop the running of your program on a particular line of code (called a breakpoint) and then examine variable values and step through subsequent lines of code.

5) Figure out how to use your debugger. Place a breakpoint on the first line of Problem 1 and run the program with the debugger. Step through the lines of code noting when variable values change.