[0404] Programmer-Self vs Tester-Self

As a programmer, you need to become schizophrenic. You need to have two different personalities AND you need to hide those two personalities from each other!

I'm talking about your "programmer-self" vs. your "tester-self."

Programming involves testing you code. Let me repeat that more forcefully: Programming requires you to test your code. Some would argue that the best programmers are really just good programmers who are really good at testing their code.

The problem is that our "programmer-self" HATES testing code because that implies that the code isn't perfect - which programmer-self is SURE that it is!

What that means is that "programmer-self" will cause us to rush through testing our code. "WHAT A WASTE OF TIME!" programmer-self is constantly saying to our sub-conscious. This leads to tons of bugs not being discovered before they cause trouble. Remember, the earlier in the development cycle a bug is found, the less expensive it is in terms of time, money, customers, etc.

So how can programmers find their bugs sooner? The first step is to find your "tester-self" personality inside of you and bring it out. Lock "programmer-self" away inside some quiet area of your brain for a while so that you can focus on testing instead.

Your "tester-self" should forget (or try to forget) all of the implementation details that you were just focused on. Instead, "tester-self" only cares about inputs and outputs. What inputs generates what outputs? Is the output correct? What if the input is wrong? Is the output still correct?

Your "tester-self" should focus on 3 types of inputs - normal range inputs, extreme inputs, invalid inputs, zero/null inputs and edge cases.

- **Normal Range Inputs** - start with what a normal user would do. What values would they input? Do those values generate good results? Normal range inputs are the LIMIT of what "programmer-self" cares about. "If normal values work, then everything must work, right??"
- **Extreme Inputs** - focuses on really big and really little values. If your program takes an integer, then it should handle both MaxInteger and MinInteger too.
- **Invalid Inputs** - if you program needs a number, type a letter. If you program needs an integer, type a decimal number. Take your keyboard and rub it all over your face - then press Enter. Not matter what input a user sends to a program, the program should NEVER crash.
- **Zero/Null Inputs** - Enter zero when asked for a number. Enter nothing when asked for a string. Make an empty list when asked for a list. Your program needs to handle all of these cases.
- **Edge Cases** - Find things in your program where the program's behavior changes. The inputs that cause those changes are called "Edge cases." If you program prints out letter grades, then there are edge cases at 100, 91, 81, 71, 61, and 0. They all need testing.

As a programmer, it is your JOB to think about all of these scenarios and test for them AT A MINIMUM. Programmers who fail to catch these kind of bugs do get fired.