# **97 Things Every Programmer Should Know**

## **Chapter 1: Act with Prudence**

1. **Before -** When given the choice between "doing it right" and "doing it quick," I choose "doing it quick" as it minimizes the remaining tasks, allows me to rest earlier, and helps me finish the project before the deadline or avoid last-minute cramming.

**After -** Acknowledge that “doing it quick” has pros and cons.

1. **Before** - “Doing it right” is time consuming.

**After** - “Doing it right” is time-consuming yet, “Doing it quick” is also time-consuming as it is harder to add new features or refactoring the program code.

1. **Before** - I don’t have any knowledge about technical debt.

**After** - Technical debt is like a loan with short-term benefits but long-term costs. Paying it off early minimizes the cost.

## **Chapter 2: Apply Functional Programming Principles**

1. **Before -** Applying Functional Programming Principles will greatly help shorten your program code, as well as an easy navigation in finding a certain function.

**After -** Functional Programming Principle is more than just shortening your programming code or for easy navigation, it will improve the quality of your code as well is improve the referential transparency.

1. **Before** - I’m not quite familiar about the Referential Transparency.

**After** - Referential Transparency should have the same output with the same input even being modified or not. This will help detect some errors or bugs and to ensure that the program code is working on its purpose.

1. **Before** - Using Functional Programming Principles is just for having a cleaner program code, and to avoid duplication of code.

**After** - Apply Functional Programming Principles will help finding the defect or error in the code, and simpler to debug.

## **Chapter 3: Ask “What Would the User Do?” (You Are not the User)**

1. **Before -** When doing our capstone project, I’m always thinking that I am user of our system.

**After -** With the title itself, we are not the user, as it will result to a bias. We think as a programmer, while user think as a user, not a programmer.

1. **Before** - Giving specific instructions to the users that gives them a big clue on how to navigate the application or system.

**After** - Giving them a big clue or specific instruction will dismiss the intention to have their pure feedback about the application such as the navigation, the user interface, and the business process.

1. **Before** - In making an application, think of a user who doesn’t know how to use a smartphone, who doesn’t have knowledge about these things.

**After** - Having their feedback is not enough to test if your application or system is functioning or user-friendly. Observing them as they navigate the system will benefit you by providing insight into their actions and the flow of your application. This will greatly help improve your user interface and its process flow.