“97 Things Every Programmer Should Know”

Chapter 71-80

Chapter 81: Two Heads Are Often Better Than One:

Working with a partner or team can lead to better ideas and solutions. Just like two heads are better than one when solving a puzzle, collaborating with others can bring fresh perspectives and help overcome challenges more effectively in software development.

Chapter 82: Two Wrongs Can Make a Right (and Are Difficult to Fix):

Combining two incorrect solutions can compound problems and make them harder to fix. It's like trying to fix a broken toy with another broken part—it might seem like a quick solution, but it often leads to more problems. It's important to address issues properly rather than trying to patch them up with quick fixes.

Chapter 83: Ubuntu Coding for Your Friends:

Writing code that's easy for others to understand and work with is important. It's like writing clear instructions for a game so your friends can play along smoothly. By prioritizing clarity and simplicity in our code, we can make collaboration easier and improve the overall quality of our software**.**

Chapter 84: The Unix Tools Are Your Friends:

Utilizing Unix tools can streamline development tasks and make them more efficient. It's like having a trusty toolbox full of helpful gadgets for fixing things around the house. By leveraging these tools, developers can automate repetitive tasks, analyze data, and troubleshoot issues more effectively.

Chapter 85: Use the Right Algorithm and Data Structure:

Choosing the right algorithm and data structure is crucial for efficient and effective software development. It's like using the right tool for the job—you wouldn't use a hammer to screw in a bolt! By selecting appropriate algorithms and data structures, developers can optimize performance and ensure their software meets its requirements.

Chapter 86: Verbose Logging Will Disturb Your Sleep:

Excessive logging in software can clutter logs and make it harder to identify important information. It's like having too many alarms going off at once—it's overwhelming and disrupts your sleep. By practicing judicious logging and focusing on relevant information, developers can maintain cleaner logs and better monitor their systems.

Chapter 87: WET Dilutes Performance Bottlenecks:

Repeated code (WET) can obscure performance issues by spreading them across multiple places. It's like having leaks in different parts of a pipe—it's harder to find the source of the problem. By refactoring duplicate code into reusable functions (DRY), developers can identify and address performance bottlenecks more effectively.

Chapter 88:When Programmers and Testers Collaborate:

Collaboration between programmers and testers leads to better software quality. It's like having two sets of eyes to check for mistakes—one for writing the code and another for testing it. By working together, developers and testers can identify issues earlier, leading to faster resolution and improved overall quality.

Chapter 89:Write Code As If You Had to Support It for the Rest of Your Life:

Writing code with longevity in mind ensures its maintainability and sustainability over time. It's like building a house with sturdy materials to withstand years of weathering. By considering future maintenance needs and potential changes, developers can write code that remains robust and adaptable throughout its lifespan.

Chapter 90 :Write Small Functions Using Examples:

Breaking down code into small, focused functions based on real-world examples improves readability and testability. It's like breaking down a complex task into smaller, manageable steps. By writing functions that perform specific tasks and providing clear examples, developers can create code that is easier to understand and maintain.

Chapter 91:Write Tests for People:

Tests should be written not only for the computer but also for the benefit of human readers. It's like writing instructions in a manual that are clear and easy to follow. By making tests readable and expressive, developers can ensure they effectively

communicate the intended behavior of the code to others, facilitating collaboration and understanding.

Chapter 92: You Gotta Care About the Code:

Taking pride in the quality of code leads to better software outcomes. It's like caring for a garden—regular attention and maintenance lead to healthier plants. By investing time and effort into writing clean, well-structured code, developers can create software that is more reliable, easier to maintain, and ultimately more satisfying to work with.

Chapter 93: Your Customers Do Not Mean What They Say:

Understanding customer needs requires digging deeper than surface-level requests. It's like listening to a friend's complaints to uncover the underlying issue. By actively listening to customers, asking probing questions, and empathizing with their goals, developers can deliver solutions that truly meet their needs and expectations.