Extracted from:

Exercises for Programmers

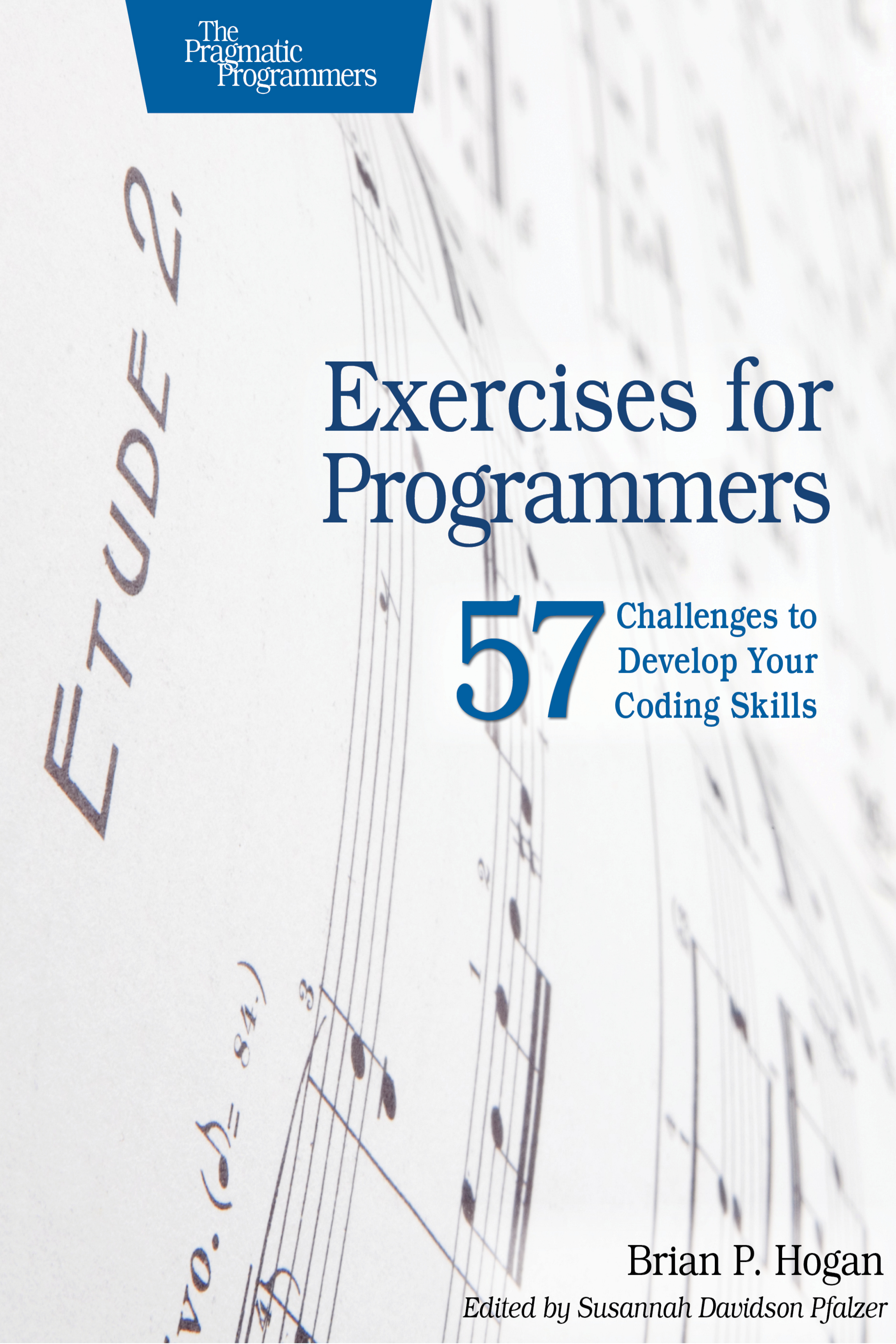
57 Challenges to Develop Your Coding Skills

Note: This extract contains some colored text (particularly in code listing). This is available only in online versions of the books. The printed versions are black and white. Pagination might vary between the online and printed versions; the content is otherwise identical.

The Pragmatic Bookshelf

Dallas, Texas • Raleigh, North Carolina

The Pragmatic Programmers



Exercises for Programmers

57 Challenges to Develop Your Coding Skills

Brian P. Hogan

Edited by Susannah Davidson Pfalzer

Exercises for Programmers

57 Challenges to Develop Your Coding Skills

Brian P. Hogan

The Pragmatic Bookshelf
Dallas, Texas • Raleigh, North Carolina

Every precaution was taken in the preparation of this book. However, the publisher assumes no responsibility for errors or omissions, or for damages that may result from the use of information (including program listings) contained herein.

The team that produced this book includes:

Susannah Davidson Pfalzer (editor)
Linda Recktenwald (copyedit)
Dave Thomas (layout)
Janet Furlow (producer)
Ellie Callahan (support)

For international rights, please contact rights@pragprog.com.

Printed in the United States of America.
ISBN-13: 978-1-68050-122-3
Encoded using the finest acid-free high-entropy binary digits.
Book version: P1.0—September 2015

How to Use This Book

Practice makes permanent.

A concert pianist practices many hours a day, learning music, practicing drills, and honing her skills. She practices the same piece of music over and over, learning every little detail to get it just right. Because when she performs, she wants to deliver a performance she is proud of for the people who spent their time and money to hear it.

A pro football player spends hours in the gym lifting, running, jumping, and doing drills over and over until he masters them. And then he practices the sport. He'll study plays and watch old game videos. And, of course, he'll play scrimmage and exhibition games to make sure he's ready to perform during the real contest.

A practitioner of karate spends a lifetime doing *kata*, a series of movements that imitate a fight or battle sequence, learning how to breathe and flex the right muscles at the right time. She may do the same series of movements thousands of times, getting better and better with each repetition.

The best software developers I've ever met approach their craft the same way. They don't go to work every day and practice on the employer's dime. They invest personal time in learning new languages and perfecting techniques in others. Of course they learn new things on the job, but because they're getting paid, there's an expectation that they are there to perform, not practice.

This book is all about practicing your craft as a programmer. Flip to a page in this book, crack open your text editor, and hammer out the program. Make your own variations on it.

Do it in a language you've never used before. And get better and better each time you do it.

Who This Book Is For

This book is targeted at two main groups of programmers.

First, it's for beginning programming students to get additional practice beyond the classroom. You can't hone your skills just by doing your assignments. Your future employer will want you to be able to demonstrate critical thinking and problem-solving skills, and you need practice to develop those. This book gives you that practice in the form of real-world problems that many developers face but that are geared toward your abilities. Each chapter covers a fundamental component of programming and is a little more complex than the previous one, building on what you've learned and preparing you for the challenges that lie ahead, both in and out of the classroom.

Many beginning programmers are used to being told exactly how to solve a problem. They often learn a language by following a written tutorial that has some code they can type. And this is a great way to start writing code. But these programmers struggle when faced with open-ended problems that don't have the solution available. And as anyone who has experience can tell you, software development is full of open-ended problems. The exercises in this book help you develop those problem-solving skills so that you build the confidence to attack even larger problems—maybe even ones that nobody else has solved yet.

But this book is also for experienced programmers looking to get better at what they do. When I learned Go and Elixir, I used programs like the ones in this book. When I tried my hand at iOS development, I tried to write these programs. And every once in a while, I do these programs in a language I already know. I'm fluent in JavaScript and Ruby, and it's a great challenge to see if I can tackle one of these programs in a different way, using a different algorithm or pattern. When I started teaching Ruby and JavaScript full time, these programs helped me discover and explain the unique features of the languages I knew how to use but didn't quite fully understand. And so if you're an experienced developer, I encourage you to do the same. Try one of these programs in Haskell. Or try to write one of these programs in every language you know and compare the results. Challenge your coworkers to do one of these exercises a week and compare your solutions. Or use these programs to mentor the new junior developer on your team.

A Note for Educators

If you teach introductory programming at the high school or college level, you may find the exercises in this book useful in your class. I don't recommend using these as summative assessments though; people reading this book are encouraged to share their solutions with others. But I do recommend using these as in-class exercises where students can work together. These exercises work well in a problem-based learning environment.

What's in This Book (And What's Not)

This book is written first and foremost to provide beginners with challenging problems they might face when first learning to program. Therefore, most of the problems are relatively simple in the beginning and gradually get more complex. The progression of exercises in this book makes practicing the fundamentals of programming challenging but fun and can accelerate the process of picking up a new language. In the first section, the programs simply take some input and manipulate the data into different output, giving you experience with how computer programs handle input and output operations. They're the kind of programs you'd do in your first week as a beginning programmer.

Next, you'll be challenged by writing programs that have you do calculations. Some of them are as simple as calculating the area of a room. But others involve financial and medical calculations similar to ones you may find on the job.

Then you'll increase the complexity of your programs by including decision logic and repetition logic, and you'll incorporate functions into them.

After that you'll find some problems that need to be solved using data structures like arrays and maps. These programs also require you to draw on some of the other problems you've solved before.

And, of course, no collection of programs would be complete without a bit of file input and output, so you'll get to practice reading data from files, processing it, and writing it back out.

Modern programs often talk with external services, so you'll find a few programs that have you work with data using third-party APIs.

Finally, a few larger programs at the end will require you to put together all the things you've learned.

In addition, each exercise includes some constraints that you'll have to follow when building the program as well as some challenges that ask you to build on the program. If you've never programmed before, you may want to skip the challenges and revisit them when you improve your skills. But if you've got some experience under your belt, you may want to accept these challenges right away if you think the program is too simple. Some of the challenges will be difficult depending on the programming language you've chosen. For example, if you're creating these programs with JavaScript and HTML, making a GUI version of the program will be easy. If you're doing this with Java, it will be a lot more work. So feel free to modify the challenges as you see fit.

However, what you *won't* find in this book are the solutions to the programs. If you think hard enough and use all of the resources at your disposal, you'll be able to figure out how to solve these problems on your own, which is the point of this book.

One last thing: you won't find the infamous interview questions here. There's no FizzBuzz. You won't need to invert binary trees, nor will you need to write a quicksort algorithm (unless you want to as part of a solution). If you're looking for things like that, you'll have to look elsewhere. Those kinds of problems have value but are often more difficult to do because it's not clear why you're doing them. That makes them unapproachable, which creates a barrier to learning.

The problems in this book are simple, real-world problems that you can easily relate to and that will help you practice solving problems with code.

What You Need

All you need is your favorite development environment—or even one you've never used. This book is programming-language agnostic. Pick a language, grab that language's reference guide, and dive in. Be warned though; the programming language you choose will determine how easy, or difficult, these programs are. For example, if you choose to do this book with Python or Ruby, then developing graphical user interfaces won't be easy. And if you choose to use JavaScript in the browser, then working with external files and web services will be much more complex than with other languages. Your approach to problems will be much different if you choose a functional programming language over an object-oriented one. But that's the real value of these exercises; they'll help you learn a language *and how that language is different from what you already know.*

You should have an Internet connection so you can do some of the programs that use third-party services and participate in the community for this book.

Online Resources

The book's website [1] has a discussion forum where you can discuss the book with other developers. Feel free to post solutions there in your favorite language and discuss your solutions with other readers. One of the most fascinating things about programming is how people approach solving problems differently and how each developer has his or her own style.

[1] http://pragprog.com/titles/bhwb