Preface

"If you really want to understand something, the best way is to try and explain it to someone else. That forces you to sort it out in your own mind... that's really the essence of programming. By the time you've sorted out a complicated idea into little steps that even a stupid machine can deal with, you've certainly learned something about it yourself." —Douglas Adams, *Dirk Gently's Holistic Detective Agency* [8]

"The world of A.D. 2014 will have few routine jobs that cannot be done better by some machine than by any human being. Mankind will therefore have become largely a race of machine tenders. Schools will have to be oriented in this direction. All the high-school students will be taught the fundamentals of computer technology, will become proficient in binary arithmetic and will be trained to perfection in the use of the computer languages that will have developed out of those like the contemporary Fortran" —Isaac Asimov 1964

I've been teaching Computer Science since 2008 and was a Teaching Assistant long before that. Before that I was a student. During that entire time I've been continually disappointed in the value (note, not quality) of textbooks, particularly Computer Science textbooks and especially introductory textbooks. Of primary concern are the costs, which have far outstripped inflation over the last decade [30] while not providing any real additional value. New editions with trivial changes are released on a regular basis in an attempt to nullify the used book market. Publishers engage in questionable business practices and unfortunately many institutions are complicit in this process.

In established fields such as mathematics and physics, new textbooks are especially questionable as the material and topics don't undergo many changes. However, in Computer Science, new languages and technologies are created and change at breakneck speeds. Faculty and students are regularly trying to give away stacks of textbooks ("Learn Java 4!," "Introduction to Cold Fusion," etc.) that are only a few years old and yet are completely obsolete and worthless. The problem is that such books have built-in obsolescence by focusing too much on technological specifics and not enough on concepts. There are dozens of introductory textbooks for Computer Science; add in the fact that there are multiple languages and many gimmicks ("Learn Multimedia Java," "Gaming with JavaScript," "Build a Robot with C!"), it is a publisher's paradise: hundreds of variations, a growing market, and customers with few alternatives.

That's why I like organizations like OpenStax (http://openstaxcollege.org/) that attempt to provide free and "open" learning materials. Though they have textbooks for a variety of disciplines, Computer Science is not one of them (currently, that is). This might be due to the fact that there are already a huge amount of resources available online such as tutorials, videos, online open courses, and even interactive code learning tools. With such a huge amount of resources, why write this textbook then? Firstly, layoff. Secondly, I don't really expect this book to have much impact beyond my own courses or department. I wanted a resource that presented an introduction to Computer Science how I teach it in my courses and it wasn't available. However, if it does find its way into another instructor's classes or into the hands of an aspiring student that wants to learn, then great!

Several years ago our department revamped our introductory courses in a "Renaissance in Computing" initiative in which we redeveloped several different "flavors" of Computer Science I (one intended for Computer Science majors, one for Computer Engineering majors, one for non-CE engineering majors, one for humanities majors, etc.). The courses are intended to be equivalent in content but have a broader appeal to those in different disciplines. The intent was to provide multiple entry points into Computer Science. Once a student had a solid foundation, they could continue into Computer Science II and pick up a second programming language with little difficulty.

This basic idea informed how I structured this book. There is a separation of concepts and programming language syntax. The first part of this book uses pseudocode with a minimum of language-specific elements. Subsequent parts of the book recapitulate these concepts but in the context of a specific programming language. This allows for a "plug-in" style approach to Computer Science: the same book could theoretically be used for multiple courses or the book could be extended by adding another part for a new language with minimal effort.

Another inspiration for the structure of this book is the Computer Science I Honors course that I developed. Usually Computer Science majors take CS1 using Java as the primary language while CE students take CS1 using C. Since the honors course consists of both majors (as well as some of the top students), I developed the Honors version to cover *both* languages at the same time in parallel. This has led to many interesting teaching moments: by covering two languages, it provides opportunities to highlight fundamental differences and concepts in programming languages. It also keeps concepts as the focus of the course emphasizing that syntax and idiosyncrasies of individual languages are only of secondary concern. Finally, actively using multiple languages in the first class provides a better opportunity to extend knowledge to other programming languages–once a student has a solid foundation in one language learning a new one should be relatively easy.

The exercises in this book are a variety of exercises I've used in my courses over the years. They have been made as generic as possible so that they could be assigned using any language. While some have emphasized the use of "real-world" exercises (whatever that means), my exercises have focused more on solving problems of a mathematical nature (most of my students have been Engineering students). Some of them are more easily understood if students have had Calculus but it is not absolutely necessary.

It may be cliché, but the two quotes above exemplify what I believe a Computer Science I course is about. The second is from Isaac Asimov who was asked at the 1964 World's Fair what he though the world of 2014 would look like. His prediction didn't become entirely true, but I do believe we are on the verge of a fundamental social change that will be caused by more and more automation. Like the industrial revolution, but on a much smaller time scale and to a far greater extent, automation will fundamentally change how we live and not work (I say "not work" because automation will very easily destroy the vast majority of today's jobs–this represents a huge economic and political challenge that will need to be addressed). The time is quickly approaching where being able to program and develop software will be considered a fundamental skill as essential as arithmetic. I hope this book plays some small role in helping students adjust to that coming world.

The first quote describes programming, or more fundamentally Computer Science and "problem solving." Computers do not *solve* problems, humans do. Computers only make it possible to automate solutions on a large scale. At the end of the day, the human race is still responsible for tending the machines and will be for some time despite what Star Trek and the most optimistic of AI advocates think.

I hope that people find this book useful. If value is a ratio of quality vs cost then this book has already succeeded in having infinite value.[1] If you have suggestions on how to improve it, please feel free to contact me. If you end up using it and finding it useful, please let me know that too!

[1]or it might be undefined, or NaN, or this book is `Exception`al depending on which language sections you read