Computer Science

AN OVERVIEW

13th Edition

J. Glenn Brookshear

(Author Emeritus)

and

Dennis Brylow

Marquette University

330 Hudson Street, NY NY 10013

A01_BROO5460_13_SE_FM.indd 1 2/6/18 5:04 PM

Senior Vice President, Courseware Portfolio Management: Engineering, Computer Science, Mathematics, Statistics, and Global Editions: Marcia Horton
Director, Portfolio Management: Engineering, Computer Science, and Global Editions: Julian Partridge
Executive Portfolio Manager: Tracy Johnson
Portfolio Management Assistant: Meghan Jacoby
Managing Producer, ECS and Mathematics: Scott Disanno
Senior Content Producer: Erin Ault
Manager, Rights and Permissions Manager: Ben Ferrini
Operations Specialist: Maura Zaldivar-Garcia
Inventory Manager: Bruce Bounty
Product Marketing Manager: Yvonne Vannatta
Field Marketing Manager: Demetrius Hall
Marketing Assistant: Jon Bryant
Full Service Project Management: Sasibalan Chidambaram, SPi Global
Composition: SPi Global
Cover Design: Black Horse Designs
Cover Photo: wacomka/Shutterstock
Cover Printer: Phoenix Color/Hagerstown
Printer/Binder: Lake Side Communications, Inc. (LSC)
Typeface: Times Ten LT Std

Copyright © 2019, 2014, 2013, 2010 by Pearson Education, Inc., Hoboken, NJ 07030. All rights reserved.

Manufactured in the United States of America. This publication is protected by copyright, and permission should be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise. For information regarding permissions, request forms and the appropriate contacts within the Pearson Education Global Rights & Permissions department, please visit www.pearsoned.com/permissions/.

Many of the designations by manufacturers and seller to distinguish their products are claimed as trademarks. Where those designations appear in this book, and the publisher was aware of a trademark claim, the designations have been printed in initial caps or all caps.

The author and publisher of this book have used their best efforts in preparing this book. These efforts include the development, research, and testing of the theories and programs to determine their effectiveness. The author and publisher make no warranty of any kind, expressed or implied, with regard to these programs or the documentation contained in this book. The author and publisher shall not be liable in any event for incidental or consequential damages in connection with, or arising out of, the furnishing, performance, or use of these programs.

Cataloging-in-Publication Data is on file at the Library of Congress

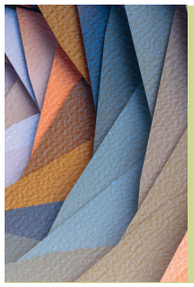
1 18

Pearson

ISBN-10: 0-13-487546-X
ISBN-13: 978-0-134-87546-0

For Dexter,
who I know will eagerly read
this book cover to cover
before he turns eight.
Oh, the places you will go . . .

A01_BROO5460_13_SE_FM.indd 3 2/6/18 5:04 PM



Contents

Chapter 0 Introduction 3

0.1 The Role of Algorithms 4
0.2 The History of Computing 6
0.3 An Outline of Our Study 11
0.4 The Overarching Themes of Computer Science 13

Chapter 1 Data Storage 25

1.1 Bits and Their Storage 26
1.2 Main Memory 34
1.3 Mass Storage 37
1.4 Representing Information as Bit Patterns 43
*1.5 The Binary System 51
*1.6 Storing Integers 56
*1.7 Storing Fractions 63
*1.8 Data and Programming 69
*1.9 Data Compression 77
*1.10 Communication Errors 83

Chapter 2 Data Manipulation 97

2.1 Computer Architecture 98
2.2 Machine Language 101
2.3 Program Execution 108
*2.4 Arithmetic/Logic Instructions 116
*2.5 Communicating with Other Devices 121
*2.6 Programming Data Manipulation 126
*2.7 Other Architectures 137

Chapter 3 Operating Systems 149

3.1 The History of Operating Systems 150
3.2 Operating System Architecture 155
3.3 Coordinating the Machine's Activities 163
*3.4 Handling Competition Among Processes 166
3.5 Security 172

Chapter 4 Networking and the Internet 183

4.1 Network Fundamentals 184
4.2 The Internet 194
4.3 The World Wide Web 206

* Asterisks indicate suggestions for optional sections.

A01_BROO5460_13_SE_FM.indd 4 2/6/18 5:04 PM

*4.4 Internet Protocols 215
*4.5 Simple Client Server 223
4.6 Cybersecurity 227

Chapter 5 Algorithms 245

5.1 The Concept of an Algorithm 246
5.2 Algorithm Representation 249
5.3 Algorithm Discovery 258
5.4 Iterative Structures 265
5.5 Recursive Structures 276
5.6 Efficiency and Correctness 285

Chapter 6 Programming Languages 305

6.1 Historical Perspective 306
6.2 Traditional Programming Concepts 317
6.3 Procedural Units 332
6.4 Language Implementation 340
6.5 Object-Oriented Programming 350
*6.6 Programming Concurrent Activities 357
*6.7 Declarative Programming 360

Chapter 7 Software Engineering 375

7.1 The Software Engineering Discipline 376
7.2 The Software Life Cycle 379
7.3 Software Engineering Methodologies 384
7.4 Modularity 387
7.5 Tools of the Trade 396
7.6 Quality Assurance 405
7.7 Documentation 408
7.8 The Human-Machine Interface 410
7.9 Software Ownership and Liability 414

Chapter 8 Data Abstractions 423

8.1 Basic Data Structures 424
8.2 Related Concepts 429
8.3 Implementing Data Structures 432
8.4 A Short Case Study 447
8.5 Customized Data Types 452
8.6 Classes and Objects 456
*8.7 Pointers in Machine Language 458

Chapter 9 Database Systems 471

9.1 Database Fundamentals 472
9.2 The Relational Model 478

*9.3 Object-Oriented Databases 489
*9.4 Maintaining Database Integrity 492
*9.5 Traditional File Structures 496
9.6 Data Mining 505
9.7 Social Impact of Database Technology 508

Chapter 10 Computer Graphics 519

10.1 The Scope of Computer Graphics 520
10.2 Overview of 3D Graphics 523
10.3 Modeling 525
10.4 Rendering 535
*10.5 Dealing with Global Lighting 547
10.6 Animation 550

Chapter 11 Artificial Intelligence 561

11.1 Intelligence and Machines 562
11.2 Perception 567
11.3 Reasoning 574
11.4 Additional Areas of Research 586
11.5 Artificial Neural Networks 593
11.6 Robotics 598
11.7 Considering the Consequences 601

Chapter 12 Theory of Computation 615

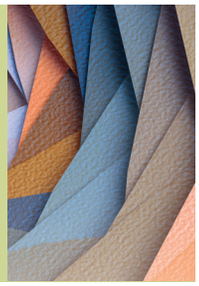
12.1 Functions and Their Computation 616
12.2 Turing Machines 619
12.3 Universal Programming Languages 623
12.4 A Noncomputable Function 629
12.5 Complexity of Problems 634
*12.6 Public-Key Cryptography 646

Appendixes 656

A ASCII 656
B Circuits to Manipulate Two's Complement Representations 657
C *Vole:* A Simple Machine Language 660
D High-Level Programming Languages 663
E The Equivalence of Iterative and Recursive Structures 665
F Answers to Questions & Exercises 667

Index 712

Preface



This book presents an introductory survey of computer science. It explores the breadth of the subject while including enough depth to convey an honest appreciation for the topics involved.

Audience

We wrote this text for students of computer science as well as students from other disciplines. As for computer science students, most begin their studies with the illusion that computer science is programming, web browsing, and Internet file sharing because that is essentially all they have seen. Yet computer science is much more than this. Beginning computer science students need exposure to the breadth of the subject in which they are planning to major. Providing this exposure is the theme of this book. It gives students an overview of computer science—a foundation from which they can appreciate the relevance and interrelationships of future courses in the field. This survey approach is, in fact, the model used for introductory courses in the natural sciences.

This broad background is also what students from other disciplines need if they are to relate to the technical society in which they live. A computer science course for this audience should provide a practical, realistic understanding of the entire field rather than merely an introduction to using the Internet or training in the use of some popular software packages. There is, of course, a proper place for that training, but this text is about educating.

While writing previous editions of this text, maintaining accessibility for nontechnical students was a major goal. The result was that the book has been used successfully in courses for students over a wide range of disciplines and educational levels, ranging from high school to graduate courses. This 13th edition is designed to continue that tradition.

New in the 13th Edition

Now in color! The move to a full color printing process in the 13th edition has allowed us to make many figures and diagrams more descriptive, and to use syntax coloring to better effect for clarifying code and pseudocode segments in the text. Most modern programming interfaces use color to aid the programmer's understanding of code; your computer science textbook should do no less.

A major theme during the development of this 13th edition has been highlighting the intersections with the new College Board Advanced Placement® Computer Science Principles ("CSP") exam. This "breadth-first" textbook for introducing computer science has included many of the big ideas and computational practices codified in the CSP framework since long before that exam came into existence; prior editions of the book have been used in

A01_BROO5460_13_SE_FM.indd 7 2/6/18 5:04 PM

pilot versions of CSP courses, and as a professional development resource for educators preparing to teach the high school version of the course. While the primary audience for this book remains college-level introductory courses, this edition explicitly calls out many points of intersection with CSP content to better assist students and instructors either preparing for the AP® CSP exam, or taking a college-level course that is intended to correspond with the credit from that exam.

The 13th edition continues the use of Python code examples and Python-like pseudocode adopted in the 12th edition. We made this change for several reasons. First, the text already contains quite a bit of code in various languages, including detailed pseudocode in several chapters. To the extent that readers are already absorbing a fair amount of syntax, it is appropriate to target that syntax toward a language they may actually see in a subsequent course. More importantly, a growing number of instructors who use this text have made the determination that even in a breadth-first introduction to computing, it is difficult for students to master many of the topics in the absence of programming tools for exploration and experimentation.

But why Python? Choosing a language is always a contentious matter, with any choice bound to upset at least as many as it pleases. Python is an excellent middle ground, with:

- a clean, easily learned syntax,
- simple I/O primitives,
- data types and control structures that correspond closely to the pseudocode primitives used in earlier editions, and
- support for multiple programming paradigms.

It is a mature language with a vibrant development community and copious online resources for further study. Python remains one of the top five most commonly used languages in the industry by some measures, and has seen a sharp increase in its usage for introductory computer science courses. It is particularly popular for introductory courses for non-majors, and has wide acceptance in other STEM fields, such as physics and biology, and as the language of choice for computational science applications.

Nevertheless, the focus of the text remains on broad computer science concepts; the Python supplements are intended to give readers a deeper taste of programming than previous editions, but not to serve as a full-fledged introduction to programming. The Python topics covered are driven by the existing structure of the text. Thus, Chapter 1 touches on Python syntax for representing data—integers, floats, ASCII, and Unicode strings. Chapter 2 touches on Python operations that closely mirror the machine primitives discussed throughout the rest of the chapter. Conditionals, loops, and functions are introduced in Chapter 5, at the time that those constructs are needed to devise a sufficiently complete pseudocode for describing algorithms. In short, Python constructs are used to reinforce computer science concepts rather than to hijack the conversation.

Every chapter has seen revisions, updates, and corrections from the previous editions.

Organization

This text follows a bottom-up arrangement of subjects that progresses from the concrete to the abstract—an order that results in a sound pedagogical presentation in which each topic leads to the next. It begins with the fundamentals of information encoding, data storage, and computer architecture (Chapters 1 and 2); progresses to the study of operating systems (Chapter 3) and computer networks (Chapter 4); investigates the topics of algorithms, programming languages, and software development (Chapters 5 through 7); explores techniques for enhancing the accessibility of information (Chapters 8 and 9); considers some major applications of computer technology via graphics (Chapter 10) and artificial intelligence (Chapter 11); and closes with an introduction to the abstract theory of computation (Chapter 12).

Although the text follows this natural progression, the individual chapters and sections are surprisingly independent and can usually be read as isolated units or rearranged to form alternative sequences of study. Indeed, the book is often used as a text for courses that cover the material in a variety of orders. One of these alternatives begins with material from Chapters 5 and 6 (Algorithms and Programming Languages) and returns to the earlier chapters as desired. I also know of one course that starts with the material on computability from Chapter 12. In still other cases, the text has been used in "senior capstone" courses where it serves as merely a backbone from which to branch into projects in different areas. Courses for less technically oriented audiences may want to concentrate on Chapters 4 (Networking and the Internet), 9 (Database Systems), 10 (Computer Graphics), and 11 (Artificial Intelligence).

On the opening page of each chapter, we have used asterisks to mark some sections as optional. These are sections that cover topics of more specific interest, or perhaps explore traditional topics in more depth. Our intention is merely to provide suggestions for alternative paths through the text. There are, of course, other shortcuts. In particular, if you are looking for a quick read, we suggest the following sequence:

Section	Topic
1.1–1.4	Basics of data encoding and storage
2.1–2.3	Machine architecture and machine language
3.1–3.3	Operating systems
4.1–4.3	Networking and the Internet
5.1–5.4	Algorithms and algorithm design
6.1–6.4	Programming languages
7.1–7.2	Software engineering
8.1–8.3	Data abstractions
9.1–9.2	Database systems
10.1–10.2	Computer graphics
11.1–11.3	Artificial intelligence
12.1–12.2	Theory of computation

There are several themes woven throughout the text. One is that computer science is dynamic. The text repeatedly presents topics in a historical

perspective, discusses the current state of affairs, and indicates directions of research. Another theme is the role of abstraction and the way in which abstract tools are used to control complexity. This theme is introduced in Chapter 0 and then echoed in the context of operating system architecture, networking, algorithm development, programming language design, software engineering, data organization, and computer graphics.

To Instructors

There is more material in this text than students can normally cover in a single semester, so do not hesitate to skip topics that do not fit your course objectives or to rearrange the order as you see fit. You will find that, although the text follows a plot, the topics are covered in a largely independent manner that allows you to pick and choose as you desire. The book is designed to be used as a course resource—not as a course definition. We suggest encouraging students to read the material not explicitly included in your course. We underrate students if we assume that we have to explain everything in class. We should be helping them learn to learn on their own.

We feel obliged to say a few words about the bottom-up, concrete-to-abstract organization of the text. As academics, we too often assume that students will appreciate our perspective of a subject—often one that we have developed over many years of working in a field. As teachers, we think we do better by presenting material from the student's perspective. This is why the text starts with data representation/storage, machine architecture, operating systems, and networking. These are topics to which students readily relate—they have most likely heard terms such as JPEG and MP3; they have probably recorded data on DVDs and flash drives; they have interacted with an operating system; and they use the Internet and smartphones daily. By starting the course with these topics, students discover answers to many of the "why" questions they have been carrying for years, and learn to view the course as practical rather than theoretical. From this beginning, it is natural to move on to the more abstract issues of algorithms, algorithmic structures, programming languages, software development methodologies, computability, and complexity, that those of us in the field view as the main topics in the science. As already stated, the topics are presented in a manner that does not force you to follow this bottom-up sequence, but we encourage you to give it a try.

We are all aware that students learn a lot more than we teach them directly, and the lessons they learn implicitly are often better absorbed than those that are studied explicitly. This is significant when it comes to "teaching" problem solving. Students do not become problem solvers by studying problem-solving methodologies. They become problem solvers by solving problems—and not just carefully posed "textbook problems." So this text contains numerous problems, a few of which are intentionally vague—meaning that there is not necessarily a single correct approach or a single correct answer. We encourage you to use these and to expand on them.

Other topics in the "implicit learning" category are those of professionalism, ethics, and social responsibility. We do not believe that this material should be presented as an isolated subject that is merely tacked on to the course. Instead, it should be an integral part of the coverage that surfaces when it is relevant. This is the approach followed in this text. You will find that Sections 3.5, 4.6, 7.9, 9.7, and 11.7 present such topics as security, privacy, liability, and social awareness in the context of operating systems, networking, software engineering, database systems, and artificial intelligence. You will also find that each chapter includes a collection of questions called Social Issues that challenge students to think about the relationship between the material in the text and the society in which they live.

Thank you for considering our text for your course. Whether you do or do not decide that it is right for your situation, I hope that you find it to be a contribution to the computer science education literature.

Pedagogical Features

This text is the product of many years of teaching. As a result, it is rich in pedagogical aids. Paramount is the abundance of problems to enhance the student's participation—over 1,000 in this 13th edition. These are classified as Questions/Exercises, Chapter Review Problems, and Social Issues. The Questions/Exercises appear at the end of each section (except for the introductory chapter). They review the material just discussed, extend the previous discussion, or hint at related topics to be covered later. These questions are answered in Appendix F.

The Chapter Review Problems appear at the end of each chapter (except for the introductory chapter). They are designed to serve as "homework" problems in that they cover the material from the entire chapter and are not answered in the text.

Also, at the end of each chapter are the questions in the Social Issues category. They are designed for thought and discussion. Many of them can be used to launch research assignments culminating in short written or oral reports.

Each chapter also ends with a list called Additional Reading that contains references to other material relating to the subject of the chapter. The websites identified in this preface, in the text, and in the sidebars of the text are also good places to look for related material.

Supplemental Resources

A variety of supplemental materials for this text are available at the book's companion website: www.pearsonhighered.com/brookshear. The following are accessible to all readers:

- Chapter-by-chapter activities that extend topics in the text and provide opportunities to explore related topics.
- Chapter-by-chapter "self-tests" that help readers to rethink the material covered in the text.
- Activities that teach the basics of Python in a pedagogical sequence compatible with the text.

In addition, the following supplements are available to qualified instructors at Pearson Education's Instructor Resource Center. Please visit www.pearsonhighered.com or contact your Pearson sales representative for information on how to access them.

- Instructor's Guide with answers to the Chapter Review Problems
- PowerPoint lecture slides
- Test bank

Errata for this book (should there be any!) will be available at http://www.mscs.mu.edu/~brylow/errata/.

To Students

Glenn Brookshear is a bit of a nonconformist (some of his friends would say more than a bit), so when he set out to write this text he didn't always follow the advice he received. In particular, many argued that certain material was too advanced for beginning students. But, we believe that if a topic is relevant, then it is relevant even if the academic community considers it to be an "advanced topic." You deserve a text that presents a complete picture of computer science—not a watered-down version containing artificially simplified presentations of only those topics that have been deemed appropriate for introductory students. Thus, we have not avoided topics. Instead, we've sought better explanations. We've tried to provide enough depth to give you an honest picture of what computer science is all about. As in the case of spices in a recipe, you may choose to skip some of the topics in the following pages, but they are there for you to taste if you wish—and we encourage you to do so.

We should also point out that in any course dealing with technology, the details you learn today may not be the details you will need to know tomorrow. The field is dynamic—that's part of the excitement. This book will give you a current picture of the subject as well as a historical perspective. With this background, you will be prepared to grow along with technology. We encourage you to start the growing process now by exploring beyond this text. Learn to learn.

Thank you for the trust you have placed in us by choosing to read our book. As authors we have an obligation to produce a manuscript that is worth your time. We hope you find that we have lived up to this obligation.

Acknowledgments

First and foremost, I thank Glenn Brookshear, who has shepherded this book, "his baby," through 11 previous editions, spanning more than a quarter century of rapid growth and tumultuous change in the field of computer science. While this is the second edition in which he has allowed a co-author to oversee all of the revisions, the pages of this 13th edition remain largely in Glenn's voice and, I hope, guided by his vision. Any new blemishes are mine; the elegant underlying framework are all his.

I join Glenn in thanking those of you who have supported this book by reading and using it in previous editions. We are honored. *Thirteen* editions for a computer science textbook? We must be nearing some kind of record.

Andrew Kuemmel (Madison West) has been an invaluable sounding board as we worked to identify the overlaps between the 13th edition and the CS Principles framework. He is the only person I know who has successfully taught many instances of the CSP course at both the high school and university levels, and his tireless advocacy for computer science educators in my home state of Wisconsin has been truly inspirational.

David T. Smith (Indiana University of Pennsylvania) played a significant role in co-authoring revisions to the 11th edition with me, many of which are still visible in this 13th edition. David's close reading of previous editions and careful attention to the supplemental materials have been essential. Andrew Kuemmel (Madison West), George Corliss (Marquette), and Chris Mayfield (James Madison) all provided valuable feedback, insight, and/or encouragement on drafts for this or previous editions, while James E. Ames (Virginia Commonwealth), Stephanie E. August (Loyola), Yoonsuck Choe (Texas A&M), Melanie Feinberg (UT-Austin), Eric D. Hanley (Drake), Sudharsan R. Iyengar (Winona State), Ravi Mukkamala (Old Dominion), and Edward Pryor (Wake Forest) all offered valuable reviews of the Python-specific revisions for the 12th edition.

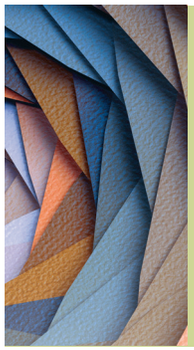
Others who have contributed in this or previous editions include J. M. Adams, C. M. Allen, D. C. S. Allison, E. Angel, R. Ashmore, B. Auernheimer, P. Bankston, M. Barnard, P. Bender, K. Bowyer, P. W. Brashear, C. M. Brown, H. M. Brown, B. Calloni, J. Carpinelli, M. Clancy, R. T. Close, D. H. Cooley, L. D. Cornell, M. J. Crowley, F. Deek, M. Dickerson, M. J. Duncan, S. Ezekiel, C. Fox, S. Fox, N. E. Gibbs, J. D. Harris, D. Hascom, L. Heath, P. B. Henderson, L. Hunt, M. Hutchenreuther, L. A. Jehn, K. K. Kolberg, K. Korb, G. Krenz, J. Kurose, J. Liu, T. J. Long, C. May, J. J. McConnell, W. McCown, S. J. Merrill, K. Messersmith, J. C. Moyer, M. Murphy, J. P. Myers, Jr., D. S. Noonan, G. Nutt, W. W. Oblitey, S. Olariu, G. Riccardi, G. Rice, N. Rickert, C. Riedesel, J. B. Rogers, G. Saito, W. Savitch, R. Schlafly, J. C. Schlimmer, S. Sells, Z. Shen, G. Sheppard, J. C. Simms, M. C. Slattery, J. Slimick, J. A. Slomka, J. Solderitsch, R. Steigerwald, L. Steinberg, C. A. Struble, C. L. Struble, W. J. Taffe, J. Talburt, P. Tonellato, P. Tromovitch, P. H. Winston, E. D. Winter, E. Wright, M. Ziegler, and one anonymous. To these individuals we give our sincere thanks.

Diane Christie designed Java and C++ manuals for the companion website in a previous edition, from which our new Python resources are descended. Thank you, Diane. Another thank you goes to Roger Eastman, who was the creative force behind the chapter-by-chapter activities that accompanied prior editions of the text, the DNA of which can still be found in the current edition's companion website activities.

My thanks to the good people at Pearson who have supported this project. Tracy Johnson, Erin Ault, Carole Snyder, and Scott Disanno in particular have brought their wisdom and many improvements to the book throughout the process.

Finally, my thanks to my wife, Petra, who distracted our three children for many long afternoons and evenings while I worked on this edition. She is my rock.

D.W.B.
Marquette University
January 01, 2018



A Correlation of *Computer Science: An Overview* 13th Edition and AP® Edition to the AP® Computer Science Principles Curriculum Framework

AP® is a trademark registered and/or owned by the College Board, which was not involved in the production of, and does not endorse, this product.

Enduring Understandings (EUs) and Learning Outcomes (LOs) from the AP® CSP Curriculum Framework are mapped to the textbook chapters and sections below. Several of the performative LOs are not explicitly covered in this text, but can be found in the supplemental resources and activities available on the companion website, www.pearsonhighered.com/brookshear.

1.1 Creative development can be an essential process for creating computational artifacts.		
1.1.1	Apply a creative development process when creating computational artifacts.	5.3
1.2 Computing enables people to use creative development processes to create computational artifacts for creative expression or to solve a problem.		
1.2.1	Create a computational artifact for creative expression.	1.0
1.2.3	Create a new computational artifact by combining or modifying existing artifacts.	10.6
1.3 Computing can extend traditional forms of human expression and experience.		
1.3.1	Use computing tools and techniques for creative expression.	10.1
2.1 A variety of abstractions built upon binary sequences can be used to represent all digital data.		
2.1.1	Describe the variety of abstractions used to represent data.	1.1, 1.4, 2.1
2.1.2	Explain how binary sequences are used to represent digital data.	1.6, 1.7, 2.1, 2.2
2.2 Multiple levels of abstraction are used to write programs or create other computational artifacts.		
2.2.1	Develop an abstraction when writing a program or creating other computational artifacts.	6.3
2.2.2	Use multiple levels of abstraction to write programs.	6.2
2.2.3	Identify multiple levels of abstractions that are used when writing programs.	1.1, 2.6, 6.1, 6.4
2.3 Models and simulations use abstraction to generate new understanding and knowledge.		
2.3.1	Use models and simulations to represent phenomena.	10.3, 10.4, 10.6
2.3.2	Use models and simulations to formulate, refine, and test hypotheses.	10.4
3.1 People use computer programs to process information to gain insight and knowledge.		
3.1.1	Find patterns, and test hypotheses about digitally processed information to gain insight and knowledge.	9.1, 9.6
3.2 Computing facilitates exploration and the discovery of connections in information.		
3.2.1	Extract information from data to discover and explain connections or trends.	9.1, 11.4
3.2.2	Determine how large data sets impact the use of computational processes to discover information and knowledge.	9.0, 9.1, 9.4
3.3 There are trade offs when representing information as digital data.		
3.3.1	Analyze how data representation, storage, security, and transmission of data involve computational manipulation of information.	1.3, 1.9, 4.4, 4.5, 9.7
4.1 Algorithms are precise sequences of instructions for processes that can be executed by a computer and are implemented using programming languages.		
4.1.1	Develop an algorithm for implementation in a program.	5.1, 5.2, 5.4, 6.3
4.1.2	Express an algorithm in a language.	5.2, 6.1

A01_BROO5460_13_SE_FM.indd 14 2/6/18 5:04 PM

4.2 Algorithms can solve many but not all computational problems.		
4.2.1	Explain the difference between algorithms that run in a reasonable time and those that do not run in a reasonable time.	12.5
4.2.2	Explain the difference between solvable and unsolvable problems in computer science.	11.3, 12.5
4.2.3	Explain the existence of undecidable problems in computer science.	12.1, 12.4
4.2.4	Evaluate algorithms analytically and empirically for efficiency, correctness, and clarity.	5.6, 12.5
5.1 Programs can be developed for creative expression, to satisfy personal curiosity, to create new knowledge, or to solve problems (to help people, organizations, or society).		
5.1.1	Develop a program for creative expression, to satisfy personal curiosity, or to create new knowledge.	10
5.1.2	Develop a correct program to solve problems.	6.2, 7.2, 7.3, 7.4, 7.7
5.1.3	Collaborate to develop a program.	5.1, 7.1, 7.6
5.2 People write programs to execute algorithms.		
5.2.1	Explain how programs implement algorithms.	2.6, 6.2
5.3 Programming is facilitated by appropriate abstractions.		
5.3.1	Use abstraction to manage complexity in programs.	1.8, 2.6, 6.3, 6.5, 8.1, 8.5
5.4 Programs are developed, maintained, and used by people for different purposes.		
5.4.1	Evaluate the correctness of a program.	1.8, 7.5
5.5 Programming uses mathematical and logical concepts.		
5.5.1	Employ appropriate mathematical and logical concepts in programming.	1.1, 1.6, 1.7, 1.8, 8.1
6.1 The Internet is a network of autonomous systems.		
6.1.1	Explain the abstractions in the Internet and how the Internet functions.	4.1, 4.2, 4.4
6.2 Characteristics of the Internet influence the systems built on it.		
6.2.1	Explain characteristics of the Internet and the systems built on it.	4.2, 4.4
6.2.2	Explain how the characteristics of the Internet influence the systems built on it.	1.3, 4.1, 4.2, 4.4, 4.6
6.3 Cybersecurity is an important concern for the Internet and the systems built on it.		
6.3.1	Identify existing cybersecurity concerns and potential options to address these issues with the Internet and the systems built on it.	4.6
7.1 Computing enhances communication, interaction, and cognition.		
7.1.1	Explain how computing innovations affect communication, interaction, and cognition.	4
7.2 Computing enables innovation in nearly every field.		
7.2.1	Explain how computing has impacted innovations in other fields.	9.6, 10.6
7.3 Computing has global effects — both beneficial and harmful — on people and society.		
7.3.1	Analyze the beneficial and harmful effects of computing.	4.6, 9.7, 11.7
7.4 Computing innovations influence and are influenced by the economic, social, and cultural contexts in which they are designed and used.		
7.4.1	Explain the connections between computing and real-world contexts, including economic, social, and cultural contexts.	4.2