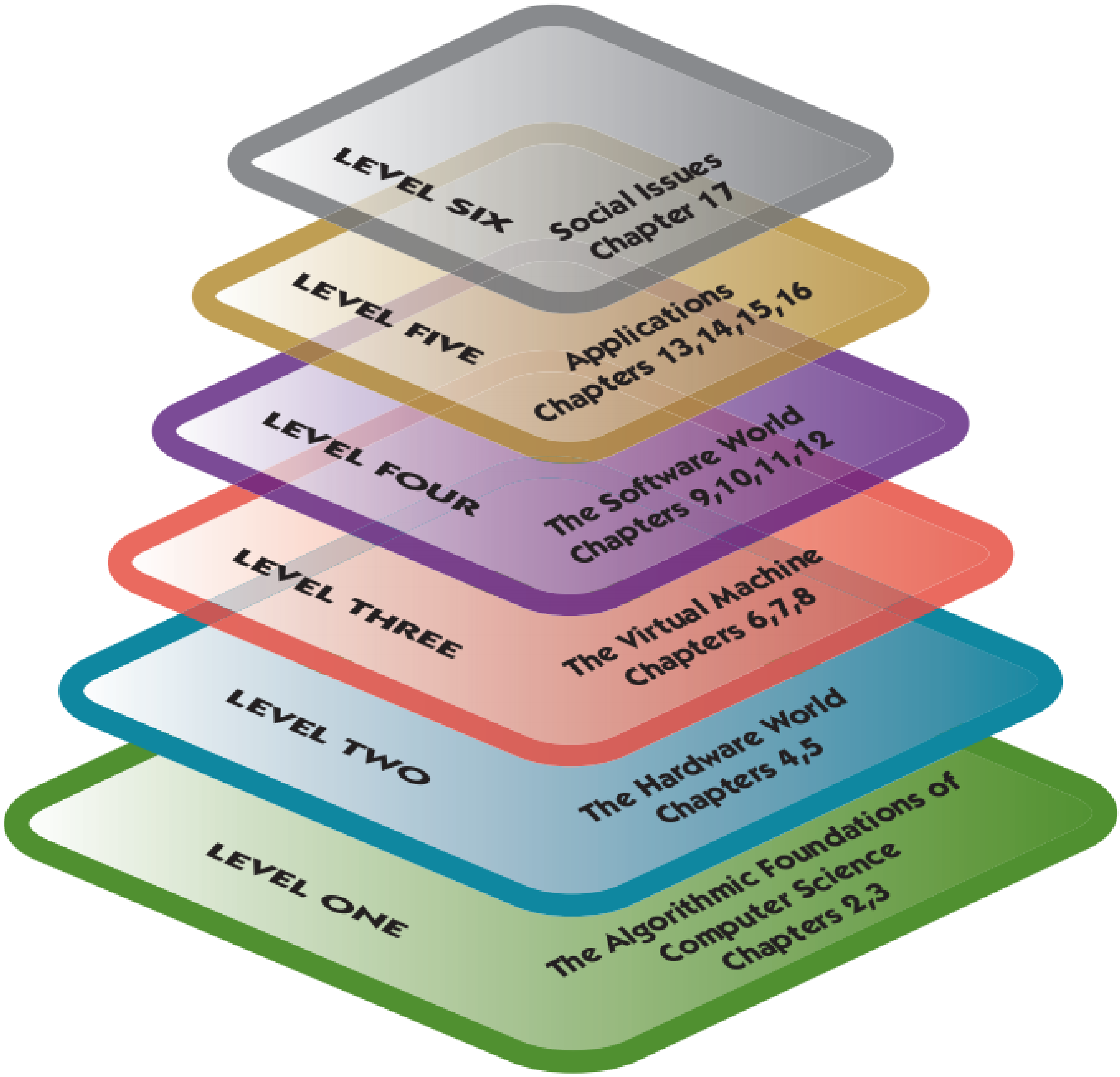
# INVITATION TO COMPUTER SCIENCE

SEVENTH EDITION

G. Michael Schneider & Judith L. Gersting

7TH EDITION

# Invitation to Computer Science


LEVEL SIX
Social Issues
Chapter 17
LEVEL FIVE
Applications
Chapters 13,14,15,16
LEVEL FOUR
The Software World
Chapters 9,10,11,12
LEVEL THREE
The Virtual Machine
Chapters 6,7,8
LEVEL TWO
The Hardware World
Chapters 4,5
LEVEL ONE
The Algorithmic Foundations of
Computer Science
Chapters 2,3

7TH EDITION

# Invitation to Computer Science

**G. Michael Schneider**
Macalester College

**Judith L. Gersting**
Indiana University-Purdue University
at Indianapolis

Contributing author:
**Bo Brinkman**
Miami University

CENGAGE Learning®

Australia • Brazil • Japan • Korea • Mexico • Singapore • Spain • United Kingdom • United States

*Invitation to Computer Science*, **Seventh Edition**

G. Michael Schneider and Judith L. Gersting

Product Director: Kathleen McMahon

Senior Content Developer: Alyssa Pratt

Development Editor: Deb Kaufmann

Marketing Manager: Eric LaScola

Manufacturing Planner: Julio Esperas

Art Director: Jack Pendleton

Production Management, Copyediting and Composition: Integra Software Services Pvt. Ltd.

Cover Photo: Studio-Pro/GettyImages



Library of Congress Control Number: 2014957763

ISBN: 978-1-305-07577-1

**Cengage Learning**
20 Channel Center Street
Boston, MA 02210
USA

Cengage Learning is a leading provider of customized learning solutions with office locations around the globe, including Singapore, the United Kingdom, Australia, Mexico, Brazil, and Japan. Locate your local office at **www.cengage.com/global.**

Cengage Learning products are represented in Canada by Nelson Education, Ltd.



To learn more about Cengage Learning Solutions, visit **www.cengage.com.**

Purchase any of our products at your local college store or at our preferred online store **www.cengagebrain.com.**

Printed in the United States of America
Print Number: 01 Print Year: 2015

*To my wife, Ruthann, our children, Benjamin, Rebecca, and Trevor, grandson, Liam, and granddaughter, Sena.*

*G. M. S.*

*To my husband, John, and to: Adam and Francine; Jason, Cathryn, Sammie, and Johnny.*

*J. L. G.*

# Brief Contents

**Online Chapters**

This text includes five language-specific online-only downloadable chapters on Ada, C++, C#, Java, and Python, available at **www.cengagebrain.com** (search for the ISBN of this book) and on the CourseMate for this text.

# Contents

## Online Chapters

This text includes five language-specific online-only downloadable chapters on Ada, C++, C#, Java, and Python, available at **www.cengagebrain.com** (search for the ISBN of this book) and on the CourseMate for this text.

# Preface to the Seventh Edition

## Overview

This text is intended for a one-semester introductory course in computer science. It presents a breadth-first overview of the discipline that assumes no prior background in computer science, programming, or mathematics. It would be appropriate for a service course for students not majoring in computer science, as well as for schools that implement their introductory course for majors using a breadth-first approach that surveys the fundamental aspects of computer science and establishes a context for subsequent courses. It would be quite suitable for a high school computer science course as well. Previous editions of this text have been used in all these types of courses.

## The Non-Majors Service Course

The introductory computer science service course (often called CS 0) has undergone numerous changes. In the 1970s and early 1980s, it was usually a class in Fortran, BASIC, or Pascal programming. In the mid-to-late 1980s, a rapid increase in computer use caused the service course to evolve into something called "computer literacy," in which students learned about new applications of computing in fields such as business, medicine, law, and education. With the growth of personal computers and productivity software, a typical early to mid-1990s version of this course would spend a semester teaching students how to use word processors, databases, spreadsheets, and email. The most recent change was its evolution into a web-centric course in which students learned to design and implement webpages using technologies such as HTML, XML, ASP, and Java applets.

In many institutions, the computer science service course is evolving once again. There are two reasons for this change. First, virtually all college and high school students are familiar with personal computers and productivity software. They have been using word processors and search engines since elementary school and are familiar with social networks, online retailing, and email; many have designed webpages and even manage their own websites and blogs. In this day and age, a course that focuses solely on applications will be of little or no interest.

But a more important reason for rethinking the structure of the CS 0 service course, and the primary reason why we authored this book, is the following observation:

> *Most computer science service courses do not teach students the foundations and fundamental concepts of computer science!*

We believe quite strongly that, in addition to applications of information technology, students in a computer science service course should receive a solid grounding in the basic concepts of the discipline. This parallels the structure of introductory courses in biology, physics, and geology, which introduce the central concepts of the fields. Topics in this type of breadth-first computer science service course would not be limited to "fun" applications such as webpage creation, social networking, game design, and interactive graphics, but would also cover foundational issues such as algorithms, hardware, computer organization, system software, language models, theory of computation, and the social and ethical issues of computing. An introduction to these core ideas exposes students to the overall richness and beauty of the field and allows them not only to use computers and software effectively but also to understand and appreciate the basic ideas underlying the discipline of computer science. As a side benefit, students who complete such a course will have a much better idea of what a major or a minor in computer science would entail.

# The First Course for Majors

Since the emergence of computer science as an academic discipline in the 1960s, the first course in the major (often called CS 1) has usually been an introduction to programming—from Fortran to BASIC to Pascal, and, later, C++, Java, and Python. But today there are numerous alternatives in the structure of CS 1, including the breadth-first overview. A first course for computer science majors using the breadth-first model emphasizes early exposure to the sub-disciplines of the field rather than placing exclusive emphasis on programming. This gives new majors a more complete and well-rounded understanding of their chosen field of study, including the many concepts and ways of thinking that are part of computer science.

Our book—intended for either majors or non-majors—is organized around this breadth-first approach as it presents a wide range of subject matter drawn from diverse areas of computer science. However, to avoid drowning students in a sea of seemingly unrelated facts and details, a breadth-first presentation must be carefully woven into a cohesive fabric, a theme, a "big picture" that ties together the individual topics and presents computer science as a unified and integrated discipline. To achieve this, our text divides the study of computer science into a hierarchy of topics, with each layer in the hierarchy building upon concepts presented in earlier chapters.

# A Hierarchy of Abstractions

The central theme of this book is that *computer science is the study of algorithms*. Our hierarchy utilizes this definition by initially looking at the algorithmic foundations of computer science and then moving upward from this central theme to higher-level issues such as hardware, systems, software, applications, and ethics. Just as the chemist starts from the basic building blocks of protons, neutrons, and electrons and then builds upon these concepts to form atoms, molecules, and compounds, so, too, does our text build from elementary concepts to higher-level ideas.

The six levels in our computer science hierarchy are:

**Level 1.** The Algorithmic Foundations of Computer Science
**Level 2.** The Hardware World
**Level 3.** The Virtual Machine
**Level 4.** The Software World
**Level 5.** Applications
**Level 6.** Social Issues in Computing

Following an introductory chapter, **Level 1** (Chapters 2–3) introduces "The Algorithmic Foundations of Computer Science," the bedrock on which all other aspects of the discipline are built. It presents fundamental ideas such as the design of algorithms, algorithmic problem solving, abstraction, pseudocode, and iteration and illustrates these ideas using well-known examples. It also introduces the concepts of algorithm efficiency and asymptotic growth and demonstrates that not all algorithms are, at least in terms of running time, created equal.

The discussions in Level 1 assume that our algorithms are executed by something called a "computing agent," an abstract concept for any entity that can carry out the instructions in our solution. However, in **Level 2** (Chapters 4–5), "The Hardware World," we now want our algorithms to be executed by "real" computers to produce "real" results. Thus begins our discussion of hardware, logic design, and computer organization. The initial discussion introduces the basic building blocks of computer systems—binary numbers, Boolean logic, gates, and circuits. It then shows how these elementary concepts can be combined to construct a real computer using the Von Neumann architecture, composed of processors, memory, and input/output. This level presents a simple machine language instruction set and explains how the algorithmic primitives of Level 1, such as assignment and conditional, can be implemented in machine language and run on the Von Neumann hardware of Level 2, conceptually tying together these two areas. It ends with a discussion of important new directions in hardware design—multicore processors and massively parallel machines.

By the end of Level 2 students have been introduced to basic concepts in logic design and computer organization, and they can appreciate the complexity inherent in these ideas. This complexity is the motivation for the material contained in **Level 3** (Chapters 6–8), "The Virtual Machine." This section describes how system software can create a more friendly, user-oriented problem-solving environment that hides many of the ugly hardware details just described. Level 3 looks at the same problems discussed in Level 2, encoding and executing algorithms, but shows how this can be done much more easily in a virtual environment containing helpful tools like a graphical user interface, editors, translators, file systems, and debuggers. This section discusses the services and responsibilities of the operating system and how it has evolved. It investigates one of the most important virtual environments in current use—networks of computers. It shows how technologies such as Ethernet, the Internet, and the web link together independent systems via transmission media and communications software. This creates a virtual environment in which we seamlessly and transparently use not only the computer on our desk or in our hand but also computing devices located anywhere in the world. This transparency has progressed to the point where we can now use systems located "in the cloud" without regard for where they are, how they provide their services, and even whether they exist as real physical entities. Level 3 concludes with a look at one of the most important services provided by a virtual machine, namely information security, and describes algorithms for protecting the user and the system from accidental or malicious damage.

Once we have created this powerful user-oriented virtual environment, what do we want to do with it? Most likely we want to write programs to solve interesting problems. This is the motivation for **Level 4** (Chapters 9–12), "The Software World." Although this book should not be viewed as a programming text, it contains an overview of the features found in modern procedural programming languages. This gives students an appreciation for the interesting and challenging task of the computer programmer and the power of the problem-solving environment created by a modern high-level language. (Detailed introductions to five important high-level programming languages are available via online, downloadable chapters accessible through the CourseMate for this text, as well as at *www.cengagebrain.com*.) There are many different language models, so Level 4 also includes a discussion of other language types, including special-purpose languages such as SQL, HTML, and JavaScript, as well as the functional, logic, and parallel language paradigms. An introduction to the design and construction of a compiler shows how high-level languages can be translated into machine language for execution. This latter discussion ties together numerous ideas from earlier chapters, as we show how an algorithm (Level 1), expressed in a high-level language (Level 4), can be compiled and executed on a typical Von Neumann machine (Level 2) by using system software tools (Level 3). These "recurring themes" and frequent references to earlier concepts help reinforce the idea of computer science as an integrated set of topics. At the conclusion of Level 4, we introduce the idea of computability and unsolvability to show students that there are provable limits to what programs, computers, and computer science can achieve.

We now have a high-level programming environment in which it is possible to write programs to solve important problems. In **Level 5** (Chapters 13–16), "Applications," we take a look at a few important uses of computers in our modern society. There is no way to cover more than a fraction of the many applications of computers and information technology in a single section. Indeed, there is hardly a field of study or an aspect of our daily lives that has not been impacted by advances in computation and communication. Instead, we focus on a small set of interesting applications that demonstrate important concepts, tools, and techniques of computer science. This includes applications drawn from the sciences and engineering (simulation and modeling), business and finance (ecommerce, databases), the social sciences (artificial intelligence), and everyday life (computer-generated imagery, video gaming, virtual communities). Our goal is to show students that these applications are not "magic boxes" whose inner workings are totally unfathomable. Rather, they are the direct result of building upon the core concepts of computer science presented in the previous chapters. We hope these discussions encourage readers to seek out information on applications specific to their own areas of interest.

Finally, we reach the highest level of study, **Level 6** (Chapter 17), "Social Issues in Computing," which addresses the social, ethical, and legal issues raised by pervasive computer technology. This section (written by contributing author Professor Bo Brinkman of Miami University) examines issues such as the theft of intellectual property, national security concerns aggravated by information technology, and the erosion of personal privacy caused by the popularity of social networks. This chapter does not attempt to provide easy solutions to these many-faceted problems. Instead, it focuses on techniques that students can use to think about ethical issues and reach their own conclusions. Our goal in this final section is to make students aware of the enormous impact that information technology is having on our society and to give them tools for making informed decisions.

This, then, is the hierarchical structure of our text. It begins with the algorithmic foundations of the discipline and works its way from lower-level hardware concepts through virtual machine environments, high-level languages, software, and applications, to the social issues raised by computer technology. This organizational structure, along with the use of recurring themes, enables students to view computer science as a unified and coherent field of study. The material in Chapters 1–12 is intended to be covered sequentially, but the applications discussed in Chapters 13–16 can be covered in any order and the social issues in Chapter 17 can be presented at any time.

## What's New in This Edition

This seventh edition of *Invitation to Computer Science* addresses a number of emerging issues in computer science. We have added significant new material on the Internet of Things, cloud computing, embedded computing, new models of electronic commerce, personal privacy, data mining, robots, and drones.

New and updated Special Interest Boxes highlight interesting historical vignettes, new developments in computing, biographies of important people in the field, and news items showing how computing is affecting our everyday lives. There are new in-chapter Practice Problems (with answers provided at the end of the text) as well as new end-of-chapter Exercises. There are also new additions to the end-of-chapter Challenge Problems; these more complex questions can be used for longer assignments done either individually or by teams of students.

## An Interactive Experience—CourseMate

This edition offers significantly enhanced supplementary material and additional resources available online through CourseMate. CourseMate is a valuable web resource containing an ebook with highlighting and note-taking capabilities, supplementary readings for each chapter, a glossary and flashcards of key technical terms, and links to interesting articles, helpful references, and relevant videos from across the web. The CourseMate encourages a truly interactive experience with study games, objective- and application-based quizzing, and hands-on exploration projects that speak students' language. Instructors may add CourseMate to the textbook package, or students may purchase CourseMate directly through *www.cengagebrain.com*.

## An Experimental Science—Laboratory Software and Manual

Another important aspect of computer science education is the realization that, like physics, chemistry, and biology, computer science is an empirical, laboratory-based discipline in which learning comes not only from watching and listening but from doing and trying. Many ideas in computer science cannot be fully understood and appreciated until they are visualized, manipulated, and tested. Today, most computer science faculty see structured laboratories as an essential part of an introductory course. We concur, and this development is fully reflected in our approach to the material.

Associated with this text is a laboratory manual and custom-designed laboratory software that enables students to experiment with the concepts we present. The manual contains 20 laboratory experiences, closely coordinated with the main text, that cover all levels except Level 6. These labs give students the chance to observe, study, analyze, and/or modify an important idea or concept. For example, associated with Level 1 (the algorithmic foundations of computer science) are experiments that animate the algorithms in Chapters 2 and 3 and ask students to observe and discuss what is happening in these animations. There are also labs that allow students to measure the running time of these algorithms for different-sized data sets and discuss their behavior, thus providing concrete observations of an abstract concept like algorithmic efficiency. There are similar labs available for Levels 2, 3, 4, and 5 that highlight and clarify the material presented in the text.

Each of the lab manual experiments includes an explanation of how to use the software, a description of how to conduct the experiment, and problems for students to complete. For these lab projects, students can either work on their own or in teams, and the course may utilize either a closed-lab (formal, scheduled) or open-lab (informal, unscheduled) setting. The manual and software work well with all these laboratory models. The text contains "Laboratory Exercise" boxes that describe each lab and identify the point in the text where it would be most appropriate.

In this new seventh edition, the Laboratory Manual has been integrated into the CourseMate for this text. Instructors may add the CourseMate, including Lab Manual, to their textbook package; contact your sales representative for more information. Students may also purchase the CourseMate/Laboratory Manual directly through *www.cengagebrain.com*.

# Programming and Online Language Modules

Programming concepts are presented in the text in the form of a survey of the features each high-level language provides and how they differ based on the computing tasks for which they were intended. Code examples are shown only to illustrate how algorithms can be embedded into the varying syntax of different languages. For instructors who want their students to have additional programming experience, online language modules for Ada, C++, C#, Java, and Python are available. Students may download any or all of these for free by going to the CourseMate for this text or to *www.cengagebrain.com*. At the CengageBrain home page, search for the ISBN of this book (found above the bar code on the back cover). At the Invitation page, click "Free Materials/Access Now". These PDF documents can be read online, downloaded to the student's computer, or printed and read on paper. Each chapter includes language-specific practice problems and exercises.

Computer science is a young and exciting discipline, and we hope that the new material in this edition, along with the laboratory projects and online modules, will convey this feeling of excitement to your students.

# Instructor Resources

The following supplemental teaching tools are available when this book is used in a classroom setting. All supplements are available to instructors for download at *sso.cengage.com*.

## Electronic Instructor's Manual

The Instructor's Manual follows the text chapter by chapter and includes material to assist in planning and organizing an effective, engaging course. The Instructor's Manual includes Overviews, Chapter Objectives, Teaching

Tips, Quick Quizzes, Class Discussion Topics, Additional Projects, Additional Resources, and Key Terms. A sample syllabus is also available.

### Solutions

Complete solutions to chapter exercises are provided.

### Test Bank

Cengage Learning Powered by Cognero® is a flexible, online system that allows you to:

- Author, edit, and manage test bank content from multiple Cengage Learning solutions
- Create multiple test versions in an instant
- Deliver tests from your Learning Management System (LMS), your classroom, or anywhere you want

### PowerPoint Presentations

Microsoft PowerPoint slides to accompany each chapter are available. Slides may be used to guide classroom presentation or to print as classroom handouts, or they may be made available to students for chapter review. Instructors may customize the slides to best suit their course with the complete Figure Files from the text.

## Acknowledgments

The authors would like to thank Alyssa Pratt and Deb Kaufmann for their invaluable assistance in developing this new edition, as well as the reviewers for this edition, whose comments were very helpful.

Colt Mazeau, *American National University*
William Oblitey, *Indiana University of Pennsylvania*
Gary Smith, *Elmhurst College*
Michael Verdicchio, *The Citadel*

Any errors, of course, are the fault solely of the authors.

—G. Michael Schneider
Macalester College
schneider@macalester.edu

—Judith L. Gersting
Indiana University-Purdue University at Indianapolis
gersting@iupui.edu