# INVITATION TO COMPUTER SCIENCE 8TH EDITION

G. MICHAEL SCHNEIDER
JUDITH L. GERSTING

8TH EDITION

# Invitation to Computer Science

**G. Michael Schneider**
Macalester College

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

CENGAGE

Australia • Brazil • Mexico • Singapore • United Kingdom • United States

**Invitation to Computer Science, 8th Edition**

**G. Michael Schneider & Judith L. Gersting**

SVP, GM Skills & Global Product Management: Jonathan Lau

Product Director: Lauren Murphy

Product Team Manager: Kristin McNary

Product Manager: Kate Mason

Executive Director, Development: Marah Bellegarde

Senior Content Development Manager: Leigh Hefferon

Developmental Editor: Deb Kaufmann

Senior Content Developer, Media: Michelle Ruelos Cannistraci

Project Manager: Ann Loch

Product Assistant: Jake Toth

Vice President, Marketing Services: Jennifer Ann Baker

Marketing Manager: Stephanie Albracht

Senior Content Project Manager: Jennifer Feltri-George

Content Digitization Project Manager: Laura Ruschman

Senior Digital Project Manager: Noah Vincelette

Senior Art Director: Diana Graham

Cover Designer: Angela Sheehan

Cover image(s): sumkinn/Shutterstock.com; sumkinn/Shutterstock.com; Studiojumpee/Shutterstock.com

Production Service/Composition: SPi Global



Library of Congress Control Number: 2017955994

Student Edition ISBN: 978-1-3375-6191-4

Loose Leaf ISBN: 978-1-337-68593-1

**Cengage**

20 Channel Center Street

Boston, MA 02210

USA

Cengage is a leading provider of customized learning solutions with employees residing in nearly 40 different countries and sales in more than 125 countries around the world. Find your local representative at **www.cengage.com.**

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

To learn more about Cengage platforms and services, visit **www.cengage.com.**

To register or access your online learning solution or purchase materials for your course, visit **www.cengagebrain.com.**

**Notice to the Reader**

Publisher does not warrant or guarantee any of the products described herein or perform any independent analysis in connection with any of the product information contained herein. Publisher does not assume, and expressly disclaims, any obligation to obtain and include information other than that provided to it by the manufacturer. The reader is expressly warned to consider and adopt all safety precautions that might be indicated by the activities described herein and to avoid all potential hazards. By following the instructions contained herein, the reader willingly assumes all risks in connection with such instructions. The publisher makes no representations or warranties of any kind, including but not limited to, the warranties of fitness for particular purpose or merchantability, nor are any such representations implied with respect to the material set forth herein, and the publisher takes no responsibility with respect to such material. The publisher shall not be liable for any special, consequential, or exemplary damages resulting, in whole or part, from the readers' use of, or reliance upon, this material.

Printed in the United States of America
Print Number: 01 Print Year: 2018

*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


Chapter 1 An Introduction to Computer Science 2

**LEVEL 1** The Algorithmic Foundations of Computer Science 42

Chapter 2 Algorithm Discovery and Design 44

Chapter 3 The Efficiency of Algorithms 92

**LEVEL 2** The Hardware World 150

Chapter 4 The Building Blocks: Binary Numbers, Boolean Logic, and Gates 152

Chapter 5 Computer Systems Organization 222

**LEVEL 3** The Virtual Machine 278

Chapter 6 An Introduction to System Software and Virtual Machines 280

Chapter 7 Computer Networks and Cloud Computing 336

Chapter 8 Information Security 394

**LEVEL 4** The Software World 432

Chapter 9 Introduction to High-Level Language Programming 434

**Chapter 10** The Tower of Babel: Programming Languages 480
**Chapter 11** Compilers and Language Translation 542
**Chapter 12** Models of Computation 588

**LEVEL 5** Applications 636

**Chapter 13** Simulation and Modeling 638
**Chapter 14** Ecommerce, Databases, and Data Science 670
**Chapter 15** Artificial Intelligence 712
**Chapter 16** Computer Graphics and Entertainment: Movies, Games, and Virtual Communities 758

**LEVEL 6** Social Issues in Computing 790

**Chapter 17** Making Decisions about Computers, Information, and Society 792

Answers to Practice Problems 833
Index 877

## Online Chapters

This text includes five language-specific online-only downloadable chapters on Ada, C++, C#, Java, and Python, available on the companion site for this text (*www.cengage.com*) and in MindTap.

# Contents


Preface to the Eighth Edition xix

**Chapter 1** An Introduction to Computer Science 2

**1.1** Introduction 2

*Special Interest Box*: In the Beginning . . . 5

**1.2** The Definition of Computer Science 6

*Special Interest Box*: Abu Ja'far Muhammad ibn Musa Al-Khwarizmi (AD 780–850?) 10

**1.3** Algorithms 12

**1.3.1** The Formal Definition of an Algorithm 12

**1.3.2** The Importance of Algorithmic Problem Solving 17

PRACTICE PROBLEMS 18

**1.4** A Brief History of Computing 18

**1.4.1** The Early Period: Up to 1940 18

*Special Interest Box*: The Original "Technophobia" 22

*Special Interest Box*: Charles Babbage (1791–1871) Ada Augusta Byron, Countess of Lovelace (1815–1852) 24

**1.4.2** The Birth of Computers: 1940–1950 24

*Special Interest Box*: John Von Neumann (1903–1957) 28

**1.4.3** The Modern Era: 1950 to the Present 28

*Special Interest Box*: And the Verdict Is . . . 29

*Special Interest Box*: The World's First Microcomputer 31

**1.5** Organization of the Text 34

LABORATORY EXPERIENCE 1 38

EXERCISES 39

CHALLENGE WORK 41

**LEVEL 1** The Algorithmic Foundations of Computer Science 42

**Chapter 2** Algorithm Discovery and Design 44

**2.1** Introduction 44

**2.2** Representing Algorithms 44

**2.2.1** Pseudocode 44

**2.2.2** Sequential Operations 48

**2.2.3** Conditional and Iterative Operations 50

PRACTICE PROBLEMS 51
*Special Interest Box*: From Little Primitives Mighty Algorithms Grow 60
2.3 Examples of Algorithmic Problem Solving 60
2.3.1 Example 1: Go Forth and Multiply 60
PRACTICE PROBLEMS 61
PRACTICE PROBLEMS 64
2.3.2 Example 2: Looking, Looking, Looking 65
LABORATORY EXPERIENCE 2 70
2.3.3 Example 3: Big, Bigger, Biggest 70
PRACTICE PROBLEMS 76
LABORATORY EXPERIENCE 3 76
2.3.4 Example 4: Meeting Your Match 77
*Special Interest Box*: Hidden Figures 84
2.4 Conclusion 84
PRACTICE PROBLEMS 85
EXERCISES 86
CHALLENGE WORK 89

Chapter 3 The Efficiency of Algorithms 92
3.1 Introduction 92
3.2 Attributes of Algorithms 92
PRACTICE PROBLEMS 97
3.3 Measuring Efficiency 97
3.3.1 Sequential Search 97
3.3.2 Order of Magnitude—Order *n* 100
*Special Interest Box*: Flipping Pancakes 102
3.3.3 Selection Sort 102
PRACTICE PROBLEM 103
PRACTICE PROBLEMS 109
3.3.4 Order of Magnitude—Order $n^2$ 109
*Special Interest Box*: The Tortoise and the Hare 113
LABORATORY EXPERIENCE 4 114
PRACTICE PROBLEM 115
3.4 Analysis of Algorithms 115
3.4.1 Data Cleanup Algorithms 115
3.4.2 Binary Search 123
PRACTICE PROBLEMS 123
PRACTICE PROBLEMS 129
LABORATORY EXPERIENCE 5 130
3.4.3 Pattern Matching 130
3.4.4 Summary 131
PRACTICE PROBLEM 132
3.5 When Things Get Out of Hand 132
PRACTICE PROBLEMS 137
3.6 Summary of Level 1 137

LABORATORY EXPERIENCE 6 138
EXERCISES 139
CHALLENGE WORK 149

**LEVEL 2** The Hardware World 150

**Chapter 4** The Building Blocks: Binary Numbers, Boolean Logic, and Gates 152

**4.1** Introduction 152
**4.2** The Binary Numbering System 153
**4.2.1** Binary Representation of Numeric and Textual Information 153
*Special Interest Box*: A Not So Basic Base 158
PRACTICE PROBLEMS 166
**4.2.2** Binary Representation of Sound and Images 167
PRACTICE PROBLEMS 175
**4.2.3** The Reliability of Binary Representation 176
**4.2.4** Binary Storage Devices 177
*Special Interest Box*: Moore's Law and the Limits of Chip Design 182
**4.3** Boolean Logic and Gates 183
**4.3.1** Boolean Logic 183
PRACTICE PROBLEMS 187
**4.3.2** Gates 188
*Special Interest Box*: George Boole (1815–1864) 192
**4.4** Building Computer Circuits 193
**4.4.1** Introduction 193
**4.4.2** A Circuit Construction Algorithm 195
PRACTICE PROBLEMS 199
**4.4.3** Examples of Circuit Design and Construction 200
LABORATORY EXPERIENCE 7 200
LABORATORY EXPERIENCE 8 208
PRACTICE PROBLEMS 209
*Special Interest Box*: Dr. William Shockley (1910–1989) 209
**4.5** Control Circuits 211
**4.6** Conclusion 215
EXERCISES 217
CHALLENGE WORK 220

**Chapter 5** Computer Systems Organization 222

**5.1** Introduction 222
**5.2** The Components of a Computer System 225
**5.2.1** Memory and Cache 227

*Special Interest Box*: Powers of 10 230
5.2.2 Input/Output and Mass Storage 238
PRACTICE PROBLEMS 239
PRACTICE PROBLEMS 244
5.2.3 The Arithmetic/Logic Unit 245
5.2.4 The Control Unit 249
PRACTICE PROBLEMS 256
5.3 Putting the Pieces Together—the Von Neumann Architecture 258
*Special Interest Box*: An Alphabet Soup of Speed Measures: MHz, GHz, MIPS, and GFLOPS 264
LABORATORY EXPERIENCE 9 265
5.4 Non–Von Neumann Architectures 265
*Special Interest Box*: Speed to Burn 269
5.5 Summary of Level 2 271
*Special Interest Box*: Quantum Computing 272
EXERCISES 273
CHALLENGE WORK 276

**LEVEL 3 The Virtual Machine 278**

**Chapter 6** An Introduction to System Software and Virtual Machines 280
6.1 Introduction 280
6.2 System Software 282
6.2.1 The Virtual Machine 282
6.2.2 Types of System Software 284
6.3 Assemblers and Assembly Language 286
6.3.1 Assembly Language 286
PRACTICE PROBLEMS 294
6.3.2 Examples of Assembly Language Code 295
PRACTICE PROBLEMS 299
LABORATORY EXPERIENCE 10 300
6.3.3 Translation and Loading 300
PRACTICE PROBLEMS 307
6.4 Operating Systems 308
6.4.1 Functions of an Operating System 308
*Special Interest Box*: A Machine for the Rest of Us 311
PRACTICE PROBLEMS 315
6.4.2 Historical Overview of Operating Systems Development 318
*Special Interest Box*: Now That's Big! 320
6.4.3 The Future 327
*Special Interest Box*: Gesture-Based Computing 330
EXERCISES 331
CHALLENGE WORK 334

**Chapter 7** Computer Networks and Cloud Computing 336
7.1 Introduction 336
7.2 Basic Networking Concepts 338
7.2.1 Communication Links 338
*Special Interest Box*: The Internet of Things 345
PRACTICE PROBLEMS 346
7.2.2 Local Area Networks 346
PRACTICE PROBLEMS 349
7.2.3 Wide Area Networks 349
7.2.4 Overall Structure of the Internet 351
*Special Interest Box*: Firewalls 354
7.3 Communication Protocols 356
7.3.1 Physical Layer 357
7.3.2 Data Link Layer 358
PRACTICE PROBLEMS 362
7.3.3 Network Layer 363
*Special Interest Box*: I Can't Believe We've Run Out 364
7.3.4 Transport Layer 366
PRACTICE PROBLEMS 367
7.3.5 Application Layer 371
7.4 Network Services and Benefits 374
LABORATORY EXPERIENCE 11 375
7.4.1 Interpersonal Communications 375
7.4.2 Social Networking 376
7.4.3 Resource Sharing 376
7.4.4 Electronic Commerce 378
7.5 Cloud Computing 379
7.6 A History of the Internet and the World Wide Web 382
7.6.1 The Internet 382
7.6.2 The World Wide Web 387
*Special Interest Box*: Geography Lesson 388
*Special Interest Box*: Net Neutrality 389
7.7 Conclusion 390
EXERCISES 390
CHALLENGE WORK 393

**Chapter 8** Information Security 394
8.1 Introduction 394
8.2 Threats and Defenses 395
8.2.1 Authentication and Authorization 396
*Special Interest Box*: The Metamorphosis of Hacking 397
PRACTICE PROBLEMS 401
8.2.2 Threats from the Network 402
*Special Interest Box*: Beware the Trojan Horse 403
*Special Interest Box*: Your Money or Your Files 404

*Special Interest Box*: Defense against the Dark Arts 406
PRACTICE PROBLEM 407
8.2.3 White Hats vs. Black Hats 407
8.3 Encryption 407
*Special Interest Box*: You've Been Hacked 408
8.3.1 Encryption Overview 409
8.3.2 Simple Encryption Algorithms 410
PRACTICE PROBLEMS 412
LABORATORY EXPERIENCE 12 413
8.3.3 DES 413
*Special Interest Box*: Hiding in Plain Sight 413
8.3.4 Public-Key Systems 417
*Special Interest Box*: Quantum Computing vs. RSA 419
PRACTICE PROBLEM 419
8.4 Web Transmission Security 420
8.5 Embedded Computing 422
*Special Interest Box*: Mischief-Makers in the Internet of Things 425
8.6 Conclusion 425
8.7 Summary of Level 3 426
EXERCISES 427
CHALLENGE WORK 429

**LEVEL 4** The Software World 432

**Chapter 9** Introduction to High-Level Language Programming 434
9.1 The Language Progression 434
9.1.1 Where Do We Stand and What Do We Want? 435
9.1.2 Getting Back to Binary 438
9.2 A Family of Languages 439
*Special Interest Box*: Ada, C++, C#, Java, and Python Online Chapters 439
9.3 Two Examples in Five-Part Harmony 440
9.3.1 Favorite Number 440
9.3.2 Data Cleanup (Again) 444
9.4 Feature Analysis 454
9.5 Meeting Expectations 454
9.6 The Big Picture: Software Engineering 463
9.6.1 Scaling Up 464
9.6.2 The Software Development Life Cycle 464
*Special Interest Box*: Vital Statistics for Real Code 466
9.6.3 Modern Environments 472
9.6.4 Agile Software Development 474

*Special Interest Box*: Software Engineering Failures 475
**9.7** Conclusion 476
EXERCISES 477
CHALLENGE WORK 477

## Online Chapters

This text includes five language-specific online-only downloadable chapters on Ada, C++, C#, Java, and Python, available on the companion site for this text (*www.cengage.com*) and in MindTap.

**Chapter 10** The Tower of Babel: Programming Languages 480
**10.1** Why Babel? 480
**10.2** Procedural Languages 482
**10.2.1** Plankalkül 482
**10.2.2** Fortran 483
**10.2.3** COBOL 484
*Special Interest Box*: Old Dog, New Tricks #1 485
PRACTICE PROBLEMS 486
PRACTICE PROBLEM 487
*Special Interest Box*: Uncle Sam Wants *Who*? 487
**10.2.4** C/C++ 488
PRACTICE PROBLEMS 492
**10.2.5** Ada 492
PRACTICE PROBLEM 493
**10.2.6** Java 494
PRACTICE PROBLEM 496
**10.2.7** Python 496
**10.2.8** C# and .NET 497
PRACTICE PROBLEM 497
*Special Interest Box*: The "Popularity" Contest 498
*Special Interest Box*: Old Dog, New Tricks #2 500
PRACTICE PROBLEM 501
**10.3** Special-Purpose Languages 501
**10.3.1** SQL 501
**10.3.2** HTML 502
LABORATORY EXPERIENCE 13 505
**10.3.3** JavaScript 505
*Special Interest Box*: Beyond HTML 506
*Special Interest Box*: PHP 509
PRACTICE PROBLEMS 509
**10.3.4** R 510

10.4 Alternative Programming Paradigms 513
10.4.1 Functional Programming 513
*Special Interest Box*: It's All in How You Look, Look, Look, . . . at It 518
PRACTICE PROBLEMS 519
LABORATORY EXPERIENCE 14 520
10.4.2 Logic Programming 520
PRACTICE PROBLEMS 525
10.4.3 Parallel Programming 526
*Special Interest Box*: New Dogs, New Tricks 531
PRACTICE PROBLEMS 532
10.5 New Languages Keep Coming 532
10.5.1 Go 532
*Special Interest Box*: Go is Going Places 533
10.5.2 Swift 534
10.5.3 Milk 535
10.6 Conclusion 535
EXERCISES 537
CHALLENGE WORK 540

**Chapter 11** Compilers and Language Translation 542
11.1 Introduction 542
11.2 The Compilation Process 545
11.2.1 Phase I: Lexical Analysis 546
11.2.2 Phase II: Parsing 550
PRACTICE PROBLEMS 550
PRACTICE PROBLEMS 556
PRACTICE PROBLEMS 567
11.2.3 Phase III: Semantics and Code Generation 568
PRACTICE PROBLEM 577
11.2.4 Phase IV: Code Optimization 577
LABORATORY EXPERIENCE 15 577
*Special Interest Box*: "Now I Understand," Said the Machine 582
11.3 Conclusion 583
EXERCISES 584
CHALLENGE WORK 587

**Chapter 12** Models of Computation 588
12.1 Introduction 588
12.2 What Is a Model? 589
12.3 A Model of a Computing Agent 591
12.3.1 Properties of a Computing Agent 591
PRACTICE PROBLEMS 592
12.3.2 The Turing Machine 593
*Special Interest Box*: Alan Turing, Brilliant Eccentric 593

PRACTICE PROBLEMS 600
12.4 A Model of an Algorithm 602
12.5 Turing Machine Examples 604
12.5.1 A Bit Inverter 605
PRACTICE PROBLEMS 607
12.5.2 A Parity Bit Machine 607
12.5.3 Machines for Unary Incrementing 610
PRACTICE PROBLEM 610
12.5.4 A Unary Addition Machine 614
PRACTICE PROBLEMS 616
LABORATORY EXPERIENCE 16 616
12.6 The Church–Turing Thesis 617
*Special Interest Box*: The Turing Award 618
12.7 Unsolvable Problems 621
*Special Interest Box*: Couldn't Do, Can't Do, Never Will Be Able to . . . 626
PRACTICE PROBLEMS 626
LABORATORY EXPERIENCE 17 627
12.8 Conclusion 627
12.9 Summary of Level 4 628
EXERCISES 629
CHALLENGE WORK 633

**LEVEL 5 Applications 636**

**Chapter 13 Simulation and Modeling 638**
13.1 Introduction 638
13.2 Computational Modeling 639
13.2.1 Introduction to Systems and Models 639
13.2.2 Computational Models, Accuracy, and Errors 642
13.2.3 An Example of Model Building 644
PRACTICE PROBLEMS 653
LABORATORY EXPERIENCE 18 654
13.3 Running the Model and Visualizing Results 654
13.4 Conclusion 664
*Special Interest Box*: The Mother of All Computations! 664
EXERCISES 665
CHALLENGE WORK 667

**Chapter 14 Ecommerce, Databases, and Data Science 670**
14.1 Introduction 670
14.2 Ecommerce 671
*Special Interest Box*: Shopping on the Web 672

14.2.1 Decisions, Decisions 673
14.2.2 Anatomy of a Transaction 675
*Special Interest Box*: A Rose by Any Other Name. . . 677
14.2.3 Designing Your Website 680
*Special Interest Box*: Less Is More 682
14.2.4 Behind the Scenes 682
PRACTICE PROBLEMS 683
14.2.5 Other Ecommerce Models 683
14.2.6 Electronic Payment Systems 685
*Special Interest Box*: Blockchain: A New Revolution? 687
14.3 Databases 688
14.3.1 Data Organization 688
14.3.2 Database Management Systems 690
14.3.3 Other Considerations 696
*Special Interest Box*: SQL, NoSQL, NewSQL 697
PRACTICE PROBLEMS 698
LABORATORY EXPERIENCE 19 699
14.4 Data Science 699
14.4.1 Tools 700
*Special Interest Box*: Algorithm Bias 703
PRACTICE PROBLEM 704
14.4.2 Personal Privacy 704
*Special Interest Box*: What Your Smartphone Photo Knows 705
14.4.3 For the Greater Good 706
14.5 Conclusion 707
EXERCISES 708
CHALLENGE WORK 711

**Chapter 15** Artificial Intelligence 712
15.1 Introduction 712
*Special Interest Box*: Victory in the Turing Test? 714
15.2 A Division of Labor 715
*Special Interest Box*: Predicted AI Milestones 718
15.3 Knowledge Representation 718
PRACTICE PROBLEMS 722
15.4 Recognition Tasks 723
*Special Interest Box*: Brain on a Chip 728
LABORATORY EXPERIENCE 20 729
PRACTICE PROBLEMS 730
15.5 Reasoning Tasks 730
15.5.1 Intelligent Searching 730
15.5.2 Swarm Intelligence 733
*Special Interest Box*: Robot Swarms 734
15.5.3 Intelligent Agents 734
15.5.4 Expert Systems 736

PRACTICE PROBLEMS 739
15.5.5 The Games We Play 739
15.6 Robots and Drones 744
15.6.1 Robots 744
*Special Interest Box*: Wait—Where Am I? 746
15.6.2 Drones 749
15.7 Conclusion 751
EXERCISES 752
CHALLENGE WORK 754

Chapter 16 Computer Graphics and Entertainment: Movies, Games, and Virtual Communities 758
16.1 Introduction 758
16.2 Computer-Generated Imagery (CGI) 761
16.2.1 Introduction to CGI 761
*Special Interest Box*: Computer Horsepower 763
16.2.2 How It's Done: The Graphics Pipeline 763
16.2.3 Object Modeling 764
16.2.4 Object Motion 767
PRACTICE PROBLEM 768
PRACTICE PROBLEM 772
16.2.5 Rendering and Display 772
16.2.6 The Future of CGI 775
16.3 Video Gaming 776
*Special Interest Box*: The Good, the Bad, and the Ugly 780
16.4 Multiplayer Games and Virtual Communities 781
16.5 Conclusion 783
*Special Interest Box*: The Computer Will See You Now 784
16.6 Summary of Level 5 785
EXERCISES 786
CHALLENGE WORK 788

LEVEL 6 Social Issues in Computing 790

Chapter 17 Making Decisions about Computers, Information, and Society 792
17.1 Introduction 792
17.2 Case Studies 793
17.2.1 Case 1: Is It Sharing or Stealing? 793
*Special Interest Box*: Death of a Dinosaur 797
PRACTICE PROBLEMS 800
*Special Interest Box*: The Sound of Music 801

17.2.2 Case 2: Legalized Snooping—Privacy vs. Security 801
*Special Interest Box*: Hero or Traitor? 803
PRACTICE PROBLEMS 809
17.2.3 Case 3: Hackers—Public Enemies or Gadflies? 809
PRACTICE PROBLEMS 815
17.2.4 Case 4: Genetic Information and Medical Research 815
*Special Interest Box*: Professional Codes of Conduct 821
17.3 Personal Privacy and Social Media 822
PRACTICE PROBLEMS 826
17.4 Fake News, Politics, and Social Media 827
17.5 Conclusion 830
17.6 Summary of Level 6 830
EXERCISES 831

*Answers to Practice Problems* 833

*Index* 877

# Preface to the Eighth Edition

## Overview

This text is intended for a one-semester introductory course in computer science. It presents a broad-based overview of the discipline that assumes no prior background in computer science, programming, or mathematics. It would be appropriate for a college or university service course for students not majoring in computer science, as well as for schools that implement their first course for majors using a breadth-first approach that surveys the fundamental aspects of computer science. It would be highly suitable for a high school computer science course, especially the AP Computer Science Principles course created by the College Board in cooperation with the National Science Foundation and colleges and universities around the United States.

## 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 teach 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 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 media, online retailing, and email; many have designed webpages and even manage their own websites and blogs. In today's world, a course that focuses on computing applications would 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 that students in a computer science service course should receive a solid grounding in the fundamental concepts of the discipline, just as introductory courses in biology, physics, and geology present the central concepts of their fields. Topics in a breadth-first computer science service course would not be limited to "fun" applications such as webpage creation, blogging, game design, and interactive graphics, but would also cover foundational issues such as algorithms, abstraction, hardware, computer organization, system software, language models, 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 and the creation of computational artifacts. 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 will entail.

This last point was the primary reason for the development of the AP Computer Science Principles high school course, which is quite similar to the breadth-first overview model just described. By learning about the field in its entirety, rather than seeing only the small slice of it called "programming," high school students will be in a better position to decide if computer science is a subject they wish to study when they begin college.

## 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, including a breadth-first overview. A first course for computer science majors using the breadth-first model emphasizes early exposure to the field's sub-disciplines rather than placing exclusive emphasis on programming. This gives new majors a complete and well-rounded understanding of the field, including the 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 coherent 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 six subareas, called layers, with each layer 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.

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

## Level 1

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.

## Level 2

However, in **Level 2** (Chapters 4–5), "The Hardware World," we 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.

## Level 3

This complexity is the motivation for the material contained in Level 3 (Chapters 6–8), "The Virtual Machine." This section describes how system software is used to create a user-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 easily in a virtual environment containing helpful tools like a graphical user interface, editors, language 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, computer networks, and 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 around 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.

## Level 4

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. (More detailed introductions to five important high-level programming languages are available via online, downloadable chapters accessible through

MindTap, as well as at *www.cengage.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, JavaScript, and R, 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) 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 insolvability to show students that there are provable limits to what programs, computers, and computer science can achieve.

## Level 5

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 some important uses of computers. There is no way to cover more than a fraction of the many applications of computers and information technology in a single section. We have included applications drawn from the sciences and engineering (simulation and modeling), business and finance (ecommerce, databases, data science), 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.

## Level 6

Finally, we reach the highest level of study, **Level 6** (Chapter 17), "Social Issues in Computing," which addresses the social, ethical, moral, and legal issues raised by pervasive computer technology. This section, based on contributions by Professor Bo Brinkman of Miami University, examines issues such as the theft of intellectual property, national security concerns, the erosion of personal privacy, and the political impact of the proliferation of fake news distributed using social media. 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 eighth edition of *Invitation to Computer Science* addresses a number of emerging issues in computer science. We have added new material on ransomware, code repositories, electronic payment systems, new programming languages such as R and Milk, data science, artificial intelligence, and drones. There is an entirely new section on fake news, politics, and social media.

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 affects our everyday lives.

## An Interactive Experience—MindTap

This edition offers significantly enhanced supplementary material and additional resources available online through MindTap. MindTap, an online teaching and learning solution, helps students be more successful and confident in the course and in their real life. MindTap guides students through the course by combining the complete textbook with interactive multimedia activities, assessments, and learning tools. Readings and activities engage students in learning core concepts, practicing needed skills, and applying what they learn. Instructors can rearrange and add content to personalize their MindTap course, and easily track students' progress with real-time analytics. MindTap integrates seamlessly with any learning management system.

## An Experimental Science—Laboratory Software and Manual

Another important aspect of computer science education is the realization that, like such scientific fields as physics, chemistry, and biology, computer science is an empirical, laboratory-based discipline in which learning comes not only from watching and listening but also 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, and this view 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 modify an important 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 lab experiment 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 eighth edition, the Laboratory Manual has been integrated into the MindTap for this text.

## 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 *www.cengage.com*. These PDF documents can be read online, downloaded to the student's computer, or printed. Each chapter includes language-specific practice problems and exercises. The exercises are also included in our educational Integrated Development Environment (IDE) within MindTap. This exposes your students to an important developer tool.

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 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 *www.cengage.com.*

## 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, Learning 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 online for instructors.

## Test Bank

Cengage Learning Testing, powered by Cognero, is a flexible, online system that allows instructors 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.

# Acknowledgments

The authors would like to thank Bo Brinkman, Ph.D., Miami University, for his contributions to the Social Issues in Computing content. The authors would also like to thank Deb Kaufmann and Emma Newsom for their invaluable assistance in developing this new edition, as well as the reviewers for this edition, whose comments were very helpful.

- Travis Dalton, Columbia College
- Debbie Collins, Black Hawk College
- Barry Poulson, University of Colorado, Boulder
- Akira Kawaguchi, The City College of New York
- H. Paul Haiduk, West Texas A&M University
- Melissa Stange, Lord Fairfax Community College
- Tom Schendl, Benedictine University

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

CHAPTER

# 1 An Introduction to Computer Science

## CHAPTER TOPICS


**1.1** Introduction

**1.2** The Definition of Computer Science

**1.3** Algorithms

**1.3.1** The Formal Definition of an Algorithm

**1.3.2** The Importance of Algorithmic Problem Solving

**1.4** A Brief History of Computing

**1.4.1** The Early Period: Up to 1940

**1.4.2** The Birth of Computers: 1940–1950

**1.4.3** The Modern Era: 1950 to the Present

**1.5** Organization of the Text

Laboratory Experience 1

EXERCISES

CHALLENGE WORK


## AFTER STUDYING THIS CHAPTER, YOU WILL BE ABLE TO:

- Understand the definition of the term algorithm
- Understand the formal definition of computer science
- Write down everyday algorithms
- Determine if an algorithm is ambiguous or not effectively computable
- Understand the roots of modern computer science in mathematics and mechanical machines
- Summarize the key points in the historical development of modern electronic computers

## 1.1 Introduction

This text is an invitation to learn about one of the youngest and most exciting scientific disciplines—*computer science*. Almost every day our newspapers, televisions, and electronic media carry reports of significant advances in computing, such as high-speed supercomputers that perform more than 90 quadrillion ($10^{15}$) mathematical operations per second; wireless networks that stream high-definition video and audio to the remotest corners of the globe in fractions of a second; minute computer chips that can be embedded into appliances, clothing, and even our bodies; and artificial intelligence systems that understand and respond to English language questions faster and more accurately than humans. The next few years will see technological breakthroughs that, until a few years ago, existed only in the minds of dreamers and science fiction writers. These are exciting times in computing, and our goal in this text is to provide you with an understanding of computer science and an appreciation for the diverse areas of research and study within this important field.

Although the average person can produce a reasonably accurate description of most scientific fields, even if he or she did not study the subject in school, many people do not have an intuitive understanding of the types of problems studied by computer science professionals. For example, you probably know that biology is the study of living organisms and that chemistry deals with the structure and composition of matter. However, you might not have the same fundamental understanding of the work that goes on in computer science. In fact, many people harbor one or more of the following common misconceptions about this field.