AP

INCLUDES

- Course framework
- Instructional section
- Sample exam questions

AP® Computer Science A

COURSE AND EXAM DESCRIPTION

Effective
Fall 2025

Introduction

Computer science involves problem-solving, hardware, and algorithms that help people utilize computers and incorporate multiple perspectives to address real-world problems in contemporary life. As the study of computer science continues to evolve, the careful design of the AP Computer Science A course strives to engage a diverse student population, including female and underrepresented students, by allowing them to discover the power of computer science through rewarding yet challenging concepts.

A well-designed, modern AP Computer Science A course that includes opportunities for students to collaborate to solve problems that interest them, as well as ones that use authentic data sources, can help address traditional issues of equity and access. Such a course can broaden participation in computing while providing a strong and engaging introduction to fundamental areas of the discipline.

The AP Computer Science A course reflects what computer science teachers, professors, and researchers have indicated are the main goals of an introductory, college-level computer science programming course:

- **Design Code**—Determine an appropriate program design and develop algorithms.
- **Develop Code**—Write and implement program code.
- **Analyze Code**—Determine the output or result of given program code or explain why code may not work as intended.
- **Document Code and Computing Systems**—Describe the behavior and conditions that produce the specified results in a program.
- **Use Computers Responsibly**—Understand the ethical and social implications of computer use.

Students practice the computer science skills of designing, developing, and analyzing their own programs to address real-world problems or pursue a passion.

Compatible Curricula

The AP Computer Science A course is compatible with the recommendations of the Association for Computing Machinery (ACM) and the Institute of Electrical and Electronics Engineers Computing Society (IEEE-CS) in the "Software Development Fundamentals" knowledge area, which is recommended for an introductory course. The AP Computer Science A course also integrates topics from the "Programming Languages" and "Algorithms and Complexity" knowledge areas. Teachers can review the **Computer Science Curricula** from ACM and IEEE-CS to see their complete curriculum guidelines.

The AP Computer Science A course vertically aligns with subconcepts in the "Algorithms and Programming" core concept in the **Computer Science Teachers Association (CSTA) K-12 Computer Science Framework**. This vertical alignment allows K-12 CS teachers to make connections from their high school courses to the college-equivalent AP Computer Science A course.

AP Computer Science Program

AP Computer Science A is one of two AP computer science courses available to students. The AP Computer Science Principles course complements AP Computer Science A by teaching the foundational areas of the discipline. Students can take these two courses in either order or concurrently, as allowed by their school.

Resource Requirements

Students should have access to a computer system that represents relatively recent technology. A school should ensure that each student has access to a computer for at least three hours a week; additional time is desirable. Student and instructor access to computers is important during class time, but additional time is essential for students to individually develop solutions to problems.

The computer system must allow students to create, edit, quickly compile, and execute Java programs comparable in size to those found in the AP Computer Science A labs. It is highly desirable that these computers provide students with access to the internet. It is essential that each computer science teacher has internet access.

A school must ensure that each student has a college-level text for individual use both inside and outside of the classroom.

Course Framework Components

Overview

This course framework provides a clear and detailed description of the course requirements necessary for student success. The framework specifies what students should know, be able to do, and understand to qualify for college credit and/or placement.

The course framework includes two essential components:

1 COMPUTATIONAL THINKING PRACTICES

The computational thinking practices are central to the study and practice of computer science. Students should develop and apply the described practices on a regular basis over the span of the course.

2 COURSE CONTENT

The course content is organized into commonly taught units of study that provide a suggested sequence for the course. These units comprise the content and conceptual understandings that colleges and universities typically expect students to be proficient in to qualify for college credit and/or placement.

1

AP COMPUTER SCIENCE A

Computational Thinking Practices

The computational thinking practices and skills for AP Computer Science A describe what a student should be able to do while exploring course concepts. The table that follows presents these practices along with their corresponding skills that students should develop during the AP Computer Science A course. These skills form the basis of tasks on the AP Computer Science A Exam. While many different skills can be applied to any one content topic, the framework supplies skill focus recommendations for each topic to help assure skill distribution and repetition throughout the course.

The unit guides that follow embed and spiral these practices throughout the course, providing teachers with one way to integrate the practices into the course content, with sufficient repetition to prepare students to apply those skills when taking the AP Computer Science A Exam.

More detailed information about teaching the computational thinking practices and skills can be found in the **Instructional Approaches** section of this publication.

Computational Thinking Practices

Practice 1	Practice 2	Practice 3	Practice 4	Practice 5
Design Code 1 Determine an appropriate program design and develop algorithms.	***Develop Code*** 2 Write and implement program code.	***Analyze Code*** 3 Determine the output or result of given program code or explain why code may not work as intended.	***Document Code and Computing Systems*** 4 Describe the behavior and conditions that produce specified results in a program.	***Use Computers Responsibly*** 5 Understand the ethical and social implications of computer use.
SKILLS				
1.A Determine an appropriate program design to solve a problem or accomplish a task. **1.B** Determine what knowledge can be extracted from data.	**2.A** Write program code to implement an algorithm. **2.B** Write program code involving data abstractions. **2.C** Write program code involving procedural abstractions.	**3.A** Determine the result or output based on statement execution order in an algorithm. **3.B** Determine the result or output based on code that contains data abstractions. **3.C** Determine the result or output based on code that contains procedural abstractions. **3.D** Explain why a code segment will not compile or work as intended and modify the code to correct the error.	**4.A** Describe the behavior of a code segment or program. **4.B** Describe the initial conditions that must be met for a code segment to work as intended or described.	**5.A** Explain how computing impacts society, economy, and culture.

2

AP COMPUTER SCIENCE A

Course Content

The AP Computer Science A course framework provides a clear and detailed description of the course requirements necessary for student success. The framework specifies what students must know, be able to do, and understand with a focus on ideas that encompass core principles, theories, and processes of computer science. The framework also encourages instruction that prepares students for advanced coursework in computer science and its integration into a wide variety of STEM-related fields, as well as for creating useful, reasonable solutions to problems encountered in a rapidly changing world.

UNITS

The course content is organized into commonly taught units. The units have been arranged in a logical sequence frequently found in many college courses and textbooks.

The four units in AP Computer Science A and their relevant weightings on the multiple-choice section of the AP Exam are listed on the following page.

Pacing recommendations on the Course at a Glance page provide suggestions for how to teach the required course content and administer the Progress Checks. The number of suggested class periods is based on a schedule in which the class meets five days a week for 45 minutes each day. While these recommendations have been made to aid planning, teachers should adjust the pacing based on the needs of their students, alternate schedules (e.g., block scheduling), or their school's academic calendar.

Return to Table of Contents

Units of Instruction	Approximate Multiple-Choice Exam Weighting
Unit 1: Using Objects and Methods	**15–25%**
Unit 2: Selection and Iteration	**25–35%**
Unit 3: Class Creation	**10–18%**
Unit 4: Data Collections	**30–40%**

Topics

Each unit is divided into teachable segments called topics. Visit the topic pages (starting on page 30) to see all the required content for each topic.

Learning Objectives and Computational Thinking Practices

In AP Computer Science A, every exam question will be aligned to a learning objective and a skill. The learning objectives represent the content domain, while the skill articulates the computational thinking practice required to successfully complete the task. The five categories of computational thinking practices are described as discrete practices; they are, in fact, interrelated and should be applied throughout the course. For a given learning objective, teachers are encouraged to ask the following questions about coding to help students apply the practices:

- How can the concepts of designing a program best be taught to students?
- Does the student's program design maximize its efficiency?
- What programs can be assigned to students that relate to their interests?
- What data can be provided to students to ensure their programs are thoroughly tested?
- What problems could a student solve by writing a computer program?
- What is the best way to teach how to debug programs?
- With all of the data that exists in various databases and websites, how can it be leveraged to help solve problems?
- How could extracting and using data be used to support a claim?

Java Quick Reference

The Java language is extensive, with far more features than could be covered in a single introductory course. The Java Quick Reference is a sheet provided to students during both the multiple-choice and free-response sections of the AP Exam. It includes a list of accessible methods from the Java library that may be included on the exam. The Java Quick Reference sheet can be found in the **Appendix** of this publication, and a copy should be given to students to use throughout the course.

Course at a Glance

Plan

The Course at a Glance provides a useful visual organization of the AP Computer Science A curricular components, including:

- Sequence of units, along with approximate weighting and suggested pacing. Please note, pacing is based on 45-minute class periods, meeting five days each week for a full academic year.
- Progression of topics within each unit.
- Computational thinking practices across units.

Teach

COMPUTATIONAL THINKING PRACTICES

- 1 Design Code
- 2 Develop Code
- 3 Analyze Code
- 4 Document Code and Computing Systems
- 5 Use Computers Responsibly

Required Course Content

Each topic contains required learning objectives and essential knowledge statements that form the basis of the assessment on the AP Exam.

Assess

Assign the Progress Checks—either as homework or in class—for each unit. Each Progress Check contains formative multiple-choice and free-response questions. The feedback from the Progress Checks shows students the areas where they need to focus.

UNIT 1 Using Objects and Methods

~32–34 Class Periods | 15–25% AP Exam Weighting

Practices	Topic
1	**1.1** Introduction to Algorithms, Programming, and Compilers
1, 2	**1.2** Variables and Data Types
2, 3	**1.3** Expressions and Output
2, 4	**1.4** Assignment Statements and Input
2, 4	**1.5** Casting and Range of Variables
3	**1.6** Compound Assignment Operators
4	**1.7** Application Program Interface (API) and Libraries
4	**1.8** Documentation with Comments
3	**1.9** Method Signatures
2, 3	**1.10** Calling Class Methods
2, 4	**1.11** `Math` Class
4	**1.12** Objects: Instances of Classes
2	**1.13** Object Creation and Storage (Instantiation)
2, 3	**1.14** Calling Instance Methods
2, 3, 4	**1.15** `String` Manipulation

Progress Check Unit 1 Part 1: Topics 1.1–1.4

Multiple-choice: ~12 questions

Progress Check Unit 1 Part 2: Topics 1.5–1.9

Multiple-choice: ~15 questions

Progress Check Unit 1 Part 3: Topics 1.10–1.15

Multiple-choice: ~18 questions
Free-response: 1 question
- Methods and Control Structures (partial)

UNIT 2 Selection and Iteration

~29–31 Class Periods | 25–35% AP Exam Weighting

Practices	Topic
1	**2.1** Algorithms with Selection and Repetition
3	**2.2** Boolean Expressions
2, 3, 4	**2.3** `if` Statements
2, 4	**2.4** Nested `if` Statements
2, 3	**2.5** Compound Boolean Expressions
2, 3	**2.6** Comparing Boolean Expressions
2, 3, 4	**2.7** `while` Loops
2, 3	**2.8** `for` Loops
2, 3, 4	**2.9** Implementing Selection and Iteration Algorithms
2, 3	**2.10** Implementing String Algorithms
2, 3, 4	**2.11** Nested Iteration
4	**2.12** Informal Run-Time Analysis

Progress Check Unit 2 Part 1: Topics 2.1–2.6

Multiple-choice: ~18 questions
Free-response: 1 question
- Methods and Control Structures (partial)

Progress Check Unit 2 Part 2: Topics 2.7–2.12

Multiple-choice: ~21 questions
Free-response: 2 questions
- Methods and Control Structures (partial)

NOTE: Partial versions of the free-response questions are provided to prepare students for more complex, full questions that they will encounter on the AP Exam.

UNIT 3 Class Creation

~20–22 Class Periods | 10–18% AP Exam Weighting

Skills	Topic
1	3.1 Abstraction and Program Design
5	3.2 Impact of Program Design
1, 2	3.3 Anatomy of a Class
2, 3	3.4 Constructors
2, 3, 4	3.5 Methods: How to Write Them
2, 4	3.6 Methods: Passing and Returning References of an Object
2, 3	3.7 Class Variables and Methods
3, 4	3.8 Scope and Access
2	3.9 `this` Keyword

Progress Check Unit 3 Part 1: Topics 3.1–3.4

Multiple-choice: ~12 questions

Progress Check Unit 3 Part 2: Topics 3.5–3.9

Multiple-choice: ~15 questions
Free-response: 2 questions

- Class Design

UNIT 4 Data Collections

~50–52 Class Periods | 30–40% AP Exam Weighting

Skills	Topic
1, 5	4.1 Ethical and Social Issues Around Data Collection
1	4.2 Introduction to Using Data Sets
2, 3	4.3 Array Creation and Access
2, 3	4.4 Array Traversals
2, 3, 4	4.5 Implementing Array Algorithms
2, 3, 4	4.6 Using Text Files
2	4.7 Wrapper Classes
2, 3	4.8 `ArrayList` Methods
2, 3, 4	4.9 `ArrayList` Traversals
2, 3, 4	4.10 Implementing `ArrayList` Algorithms
2, 3	4.11 2D Array Creation and Access
2, 3	4.12 2D Array Traversals
2, 3, 4	4.13 Implementing 2D Array Algorithms
2, 3, 4	4.14 Searching Algorithms
3, 4	4.15 Sorting Algorithms
3, 4	4.16 Recursion
4	4.17 Recursive Searching and Sorting

Progress Check Unit 4 Part 1: Topics 4.1–4.5

Multiple-choice: ~18 questions
Free-response: 2 questions

- Data Analysis with Array (partial)

Progress Check Unit 4 Part 2: Topics 4.6–4.10

Multiple-choice: ~21 questions
Free-response: 2 questions

- Data Analysis with `ArrayList`

Progress Check Unit 4 Part 3: Topics 4.11–4.17

Multiple-choice: ~21 questions
Free-response: 2 questions

- 2D Array

Artificial Intelligence in AP Computer Science A

The Value of Dual Competency: Artificial Intelligence (AI) Tools and Core Programming Skills

AI-powered tools are increasingly utilized in the field of computer science, both in the classroom and in careers. In the classroom, successful students combine strong foundational knowledge with effective AI tool usage. While AI tools can enhance abilities, their effectiveness depends on a user's knowledge and judgment.

For example, a student with solid programming fundamentals can evaluate AI-generated code for accuracy, write precise prompts, identify and fix issues, integrate AI-generated code with their own, and explore advanced concepts. Conversely, a student lacking these fundamentals might accept incorrect AI suggestions, struggle with debugging, fail to adapt AI solutions, miss errors, and have difficulty merging AI suggestions with existing code.

The AP Computer Science A course focuses on helping students build the fundamentals necessary for effective AI use. This course does not currently assess AI skills, nor are students permitted to use AI during assessments. Success in the course and on the AP Exam requires understanding and mastery of the course content; however, AI tools can be valuable learning aids, helping students deepen their comprehension and skills. Teachers can integrate AI use into their instruction to help prepare students for future computing challenges.

The following definitions, guidelines for acceptable use, and instructional scenarios can help teachers set clear expectations and incorporate AI tools into their teaching.

Acceptable Generative AI Use in AP Computer Science A

Using generative AI (GenAI) tools in software development can include code completion and generation tools (e.g., GitHub, Microsoft Copilot, Amazon CodeWhisperer), large language models (e.g., ChatGPT), and other AI-assisted development tools. These tools can generate code, explain programming concepts, debug issues, and assist with various aspects of development.

GenAI tools should be used ethically, responsibly, and intentionally to support student learning, not to bypass it. The goal is for students to become skilled problem solvers who can effectively leverage AI tools *because* of their strong fundamental understanding and independent programming abilities.

Students should recognize that while AI tools can accelerate learning and development, success in AP Computer Science A assessments requires independent mastery of course content and practices, and all AP assessments are conducted without AI assistance, making personal competency essential.

Key Principles for Using AI in AP Computer Science A:

- AI tools should enhance, not replace, the learning process.
- Students must develop independent programming and problem-solving abilities as outlined in the course framework to succeed in the course and on the exam.
- Understanding how code works is more important—and a more valuable skill—than simply generating it.
- AI can be used as a learning aid to build deeper comprehension.
- Students should develop skills to evaluate and verify AI-generated code.

Practice	Uses That Will Enhance Learning	Uses That Will Undermine Learning
Design Code	▪ Using AI to explore different approaches to solving a programming problem ▪ Getting suggestions for algorithm design and data structure selection ▪ Analyzing AI-generated solutions to understand different design patterns ▪ Using AI to break down complex problems into smaller components	▪ Relying on AI to design solutions without understanding the underlying logic ▪ Using AI-generated code without analyzing its structure and efficiency ▪ Accepting AI suggestions without considering alternative approaches ▪ Bypassing the problem-solving process by immediately asking AI for complete solutions
Develop Code	▪ Using AI for syntax assistance and basic code structure ▪ Learning from AI explanations of code concepts (e.g., constructors, methods) ▪ Getting help with debugging specific errors ▪ Using AI to understand different implementation approaches ▪ Requesting examples of specific Java concepts (e.g., array manipulation)	▪ Copying AI-generated code without understanding how it works ▪ Using AI to complete homework or labs without learning the concepts ▪ Relying on AI for all programming tasks without developing independent skills ▪ Letting AI handle the entire development process without student involvement
Analyze Code	▪ Using AI to help understand unfamiliar code ▪ Getting explanations of code behavior and execution flow ▪ Learning how to identify and fix common programming errors ▪ Practicing code review skills with AI assistance	▪ Using AI to bypass understanding program execution and behavior ▪ Relying solely on AI for debugging without learning troubleshooting skills
Document Code and Computing Systems	▪ Getting suggestions for clear and effective comments ▪ Learning best practices for documentation ▪ Understanding how to describe code behavior and purpose ▪ Using AI to solicit feedback on how to improve documentation clarity	▪ Using AI to generate documentation without understanding the code ▪ Relying on AI for all documentation without developing personal documentation skills ▪ Accepting AI-generated documentation without verifying its accuracy
Use Computers Responsibly	▪ Learning about the ethical implications of AI in software development ▪ Understanding how to verify and validate AI-generated code ▪ Developing skills to assess AI tool capabilities and limitations ▪ Using AI tools to explore industry best practices	▪ Using AI in ways that violate academic integrity ▪ Sharing or submitting AI-generated code without proper attribution ▪ Using AI tools to circumvent learning objectives ▪ Failing to develop independent problem-solving skills

Teachers can help students explore these acceptable and productive uses through structured practice and guided learning experiences. Regular formative assessments, such as the use of Progress Checks and Practice Exams without AI assistance in AP Classroom, will help ensure students are developing the independent capabilities necessary for success in this course and in their careers.

Developing the Computational Thinking Practices

Throughout the course, students will develop computational thinking practices that are fundamental to the study of computer science. Because computational thinking practices represent the complex skills demonstrated by adept computer scientists, students will benefit from multiple opportunities to develop these practices in a scaffolded manner.

All the multiple-choice questions on the AP Exam will be associated with one of the skills in the tables that follow. In addition, each scoring criteria in the scoring guidelines for the free-response questions will be associated with skills in the practice of developing code. The sample exam questions later in this publication show how the questions relate to specific computational thinking practices.

The tables that follow look at each computational thinking practice and its associated skills and provide examples of questions with sample activities for incorporating instruction on that skill into the course.

Computational Thinking Practice 1: Design Code

The table that follows provides examples of questions and sample activities for teaching students to successfully design code using different topics throughout the course.

Skill	Questions to Ask Students	Sample Activities
1.A *Determine an appropriate program design to solve a problem or accomplish a task.*	What strategy did you use to solve this problem? How does using the cues from nouns and verbs in a problem description help determine a class design? How does creating a diagram aid in the class design process?	Give students a problem specification and have them determine the appropriate classes and/or methods that need to be written. As students read through the specification, ask them to highlight nouns and circle verbs to help ascertain what classes might be needed, as well as potential attributes and behaviors of those classes. Have students create UML diagrams of existing classes and methods and refer to those diagrams when deciding what method calls are legal and appropriate.
1.B *Determine what knowledge can be extracted from data.*	Does the data set have enough information to answer your question? Is there any bias within the data set?	Direct students to the U.S. Government's Open Data site **data.gov** and have them find a data set that interests them. Ask them what analysis can be done using that set.

Computational Thinking Practice 2: Develop Code

The table that follows provides examples of questions and sample activities for teaching students to write code following algorithms that contain data abstractions and procedural abstractions that include different topics throughout the course.

Skill	Questions to Ask Students	Sample Activities
2.A *Write program code to implement an algorithm.*	Is there more than one way to implement an algorithm? What makes one algorithm a better choice than another algorithm? What line or lines of code can be used to complete a given task?	Provide students a code segment with a missing expression, and ask them what statement should be used to complete the code segment. Have them share their solution with a partner to see if they have the same answer or can come to an agreement. Then have pairs share with the entire class. Before introducing searching algorithms, have students search for a specific item among a large group of unsorted items, making note of their process and the time it takes. Then ask them to search for an item in a large group of sorted items and note the difference in process and time taken. Give students an algorithm for one of the sorts in the course (insertion, selection, or merge sort) and have them write the code to implement that sort.
2.B *Write program code involving data abstractions.*	How do you know what data type to use for your variables? Which data structure is most appropriate to use for a given problem? Should you use an array or `ArrayList` as your data structure? What are the pros and cons for using each?	Give students lists of different types of objects and have them identify the appropriate data type that should be used to store data for that object. When teaching how to create objects, model the correct way to declare a reference type variable and then initialize it with a call to a constructor. This helps students recognize the parts of the statement that change, such as the class name and parameter list, and what remains the same, such as the `new` keyword and the statement structure. Give students a problem specification and ask them to design their own class. Then have them work with a partner to see if there is any expected or desired behavior missing. Ask students to form groups of four and come up with a way of traversing a 2D array other than row-major order. Then ask them to develop and implement the algorithm to match their traversal. Create four new groups of students, comprised of one member from each of the former groups, and have them present their pattern and program code for the traversal.

continued on next page

Skill	Questions to Ask Students	Sample Activities
2.C *Write program code involving procedural abstractions.*	Should you use methods included in available Java libraries or should you design your own class and write methods for it? Does the method you are using need parameters? If so, how many and what are their types? What prompt framework could be used to produce a step-by-step walkthrough and help you learn why it is important to break down tasks? What role do you want the AI prompt framework tool to play? How will you communicate that in your prompt?	When introducing new methods to students, provide them with the method signature and have them come up with their own method call examples to help them remember how they function. Ask students to write the steps needed to define a method so they can refer to a record of the process later. Use AI explanations to build student understanding of procedural abstractions by having students request step-by-step walkthroughs of example code. Encourage them to use a high-value prompt framework to craft prompts that go beyond the walkthrough. For instance, students can be taught to ask AI to explain *why* breaking tasks into procedures or methods enhances reusability and maintainability, reinforcing both the conceptual and practical benefits of procedural abstraction.

Computational Thinking Practice 3: Analyze Code

The table that follows provides examples of questions and sample activities for teaching students to analyze different topics throughout the course.

Skill	Questions to Ask Students	Sample Activities
3.A *Determine the result or output based on statement execution order in an algorithm.*	What output does the code segment produce? What are the values of the variables as the program executes?	When teaching iteration, trace tables are helpful for keeping track of loop variables, other program variables, and output. Have students make a column for each variable in their code and a column for output. Each time through, the loop is represented by a new row in the table, with the current value of each variable being stored in cells on the table.
3.B *Determine the result or output based on code that contains data abstractions.*	How can you determine the output of a segment of code? What strategy should you use?	Students can check whether they have been given the correct output for a given method by tracing their code with input values that are around the extreme ranges of the input values.
3.C *Determine the result or output based on code that contains procedural abstractions.*	How can you best use the Java Quick Reference (see **Appendix**) to understand method calls?	When introducing students to new methods, have them predict the output of several different statements that use that method call. Then have them create a program that contains the same statements and compare actual and expected results.
3.D *Explain why a code segment will not compile or work as intended and modify the code to correct the error.*	When a method was called in the program, did you make sure it was called with the correct number and type of parameters? Are the parameters in the correct order? Is the method a void method or does it return a value? What prompt framework could be used to help you diagnose and debug your code? What kind of output do you want? Is your prompt specific enough for the AI tool to know this?	Ask students to consider a code segment that does not work as intended. Give them a few minutes to think independently about the reason why and how to fix the code. Have students compare and add to this list with a partner before sharing with the whole group. Pass out papers with various segments of code that have errors such as parameter mismatch, incorrect Boolean expressions or loop bounds, or incorrect order of expressions. Ask students to identify the errors. Use AI tools to help students develop the skill of identifying and fixing common programming errors. Have students input code containing errors into a generative AI platform and evaluate the suggested improvements. To deepen their understanding, teach students to prompt the AI not only to correct errors but also to explain *why* the original code does not compile or work as intended. This approach builds both error diagnosis and debugging skills.

Computational Thinking Practice 4: Document Code and Computing Systems

The table that follows provides examples of questions and sample activities for teaching students to describe how code is used to incorporate different topics throughout the course.

Skill	Questions to Ask Students	Sample Activities
4.A *Describe the behavior of a code segment or program.*	What is a given code segment supposed to do? What comments can be added to a given code segment so that another programmer can understand its purpose?	When covering nested conditionals and compound Boolean expressions, provide different code segments to students. Have them paraphrase what is happening in the code, instead of just explaining each line or expression individually. Then have them pair up with students whose code accomplishes the same task in a different manner. When looking at a recursive method to determine how many times it executes, have students call each other as recursive methods. Students should stand up when "called." The count would be equivalent to the number of students in the room who are standing as a result of the calls.
4.B *Describe the initial conditions that must be met for a code segment to work as intended or described.*	What parameters are needed when a method is called? Does a given method have preconditions? If so, did you make sure they were checked before the method was called?	As students begin to write methods, provide them with the specifications for several methods and a collection of method headers. Ask them to select which method header belongs with which specification, and have them implement each method. Provide students with a complete class implementation, with associated Java documentation, and have them circle the pre- and postconditions for each method.

Computational Thinking Practice 5: Use Computers Responsibly

The table that follows provides examples of questions and sample activities for teaching students to be aware of the impacts of computing throughout the course.

Skill	Questions to Ask Students	Sample Activities
5.A *Explain how computing impacts society, economy, and culture.*	How have computers changed the way people relate to one another? In what ways have computers impacted culture?	Have students interview someone from an older generation, asking questions about how they did certain tasks before computers were readily available. Each student should be prepared to share out with the whole group. As an example of learning about ethical implications of AI in software development, ask students to research how AI is impacting the job market, productivity, and innovation. Have them form groups and discuss their findings.

Using an Iterative Development Process

The instructional strategies tables include strategies that facilitate students' use of a process to plan and create computer programs. Planning processes are iterative and cyclical in nature and require students to reflect on what they have created. If necessary, students should return to prior stages to modify their plans and change their development. This iterative process of development and revision allows students to have more confidence in their solution and in their ability to develop solutions. Allowing students to revise and improve on previous work is also another way to provide feedback and can be used as a method of formative assessment throughout the course.

Several of the computational thinking practices for this course represent phases that can be seen in a development process to plan and create computer programs, including:

- **Design Code:** In this phase, students create a representation or model of their solution. Strategies such as **diagramming** and **using manipulatives** can give students confidence in the design of their solution.
- **Develop Code:** In this phase, students implement their solution by writing a program. The strategy of **pair programming** allows students to have constant support during implementation. To successfully implement a program, students need to understand the operators and program execution order needed to produce the desired results.
- **Analyze Code:** In this phase, students evaluate and test the solution they implemented. Use strategies such as **predict and confirm** and **error analysis** to determine whether the solution is appropriate. This phase may require students to revisit earlier phases of development to make improvements on their solution.
- **Document Code and Computing Systems:** In this phase, students describe what the program does, often adding comments to program code. Students could keep a journal or log book to note design decisions and rationales. This allows students to come back to a program later and recall how it was constructed. This is helpful when modifications need to be made to an existing program, either by the student who wrote the initial version or by another student. Strategies such as **marking the text** or **paraphrasing** can help students document their process at different stages.
- **Use Computers Responsibly:** In this phase, students explain how computing is an important part of our lives and with it comes great responsibility to society, the economy, and culture.

These phases need not be implemented in linear order. Students may choose to return to earlier phases as their design ideas change and develop. Throughout any design process, students should collaborate with others when developing a program.

AP COMPUTER SCIENCE A

Exam Information

Task Verbs Used in Free-Response Questions

The following task verbs are commonly used in the free-response questions:

Assume: Suppose to be the case without any proof or need to further address the condition.

Complete (program code): Express in print form the proper syntax to represent a described algorithm or program given part of the code.

Implement/Write: Express in print form the proper syntax to represent a described algorithm or program.

Sample Exam Questions

The sample exam questions that follow illustrate the relationship between the course framework and the AP Computer Science A Exam and serve as examples of the types of questions that appear on the exam. These sample questions do not represent the full range and distribution of items on an official AP Computer Science A Exam. After the sample questions is a table that shows which skill, learning objective(s), and essential knowledge statement(s) each question assesses. The table also provides the answers to the multiple-choice questions.

Section I: Multiple-Choice

1. Consider the following code segment.

```
double q = 15.0;
int r = 2;

double x = (int) (q / r);
double y = q / r;

System.out.println(x + " " + y);
```

What is printed as a result of executing this code segment?

(A) `7.0 7.0`

(B) `7.0 7.5`

(C) `7.5 7.0`

(D) `7.5 7.5`

2. Consider the following code segment.

```
int x = ((int) (Math.random() * 10)) + 5;
int y = ((int) (Math.random() * 5)) + 10;
System.out.println(x + " " + y);
```

Which of the following could be printed as a result of executing this code segment?

(A) `8 15`

(B) `9 12`

(C) `10 7`

(D) `15 11`

3. Consider the following `Date` class.

```
public class Date
{
   private String month;
   private int day;

   public Date(String m, int d)
   { /* implementation not shown */ }

   /* Class contains no other constructors.
      Other methods are not shown. */
}
```

Which of the following code segments, appearing in a class other than `Date`, will correctly create an instance of a `Date` object?

(A) `Date birthday = new Date("September", "5th");`

(B) `Date birthday = new Date("September", 5);`

(C) `Date birthday = new Date("September 5");`

(D)
```
Date birthday = new Date();
   birthday.month = "September";
   birthday.day = 5;
```

4. Consider the following code segment.

```
String str1 = "LMNOP";
String str2 = str1.substring(3);
str2 += str1.substring(2, 3);
```

What is the value of `str2` after executing this code segment?

(A) `"OPN"`

(B) `"OPNO"`

(C) `"NOPM"`

(D) `"NOPMN"`

5. The following method is intended to return `"Renaissance"` for years from 1400 to 1600, inclusive, `"Medieval"` for years from 400 to 1399, inclusive, or `"Other"` for any other year. In the method, the parameter represents a year.

```
public static String getCategory(int year)
{
   String category = "";
   /* missing code */
   return category;
}
```

Which of the following code segments can replace `/* missing code */` so that the `getCategory` method works as intended?

(A)
```
if (year > 1600 || year < 400)
{
   category = "Other";
}
else if (year >= 1400)
{
   category = "Renaissance";
}
else if (year >= 400)
{
   category = "Medieval";
}
```

(B)
```
if (year > 1600 || year < 400)
{
   category = "Other";
}
if (year >= 1400)
{
   category = "Renaissance";
}
if (year >= 400)
{
   category = "Medieval";
}
```

(C)
```
if (year >= 1400)
{
   category = "Renaissance";
}
else if (year >= 400)
{
   category = "Medieval";
}
else
{
   category = "Other";
}
```

(D)
```
if (year >= 1400)
{
   category = "Renaissance";
}
if (year >= 400)
{
   category = "Medieval";
}
else
{
   category = "Other";
}
```

6. Assume that `x`, `y`, and `z` are `boolean` variables that have been properly declared and initialized.

   Which of the following expressions is equivalent to `!x || !y || z`?

   (A) `!(x && y) && z`

   (B) `!(x && y) || z`

   (C) `!(x || y) && z`

   (D) `!(x || y) || z`

7. Consider the following code segment.

```
for (int j = 4; j > 0; j--)
{
   for (/* missing code */)
   {
      System.out.print(k + " ");
   }
   System.out.println();
}
```

   This code segment is intended to produce the following output.

```
0 1 2 3
0 1 2
0 1
0
```

   Which of the following can be used to replace `/* missing code */` so that this code segment works as intended?

   (A) `int k = 0; k < j; k++`

   (B) `int k = 0; k <= j; k++`

   (C) `int k = j; k > 0; k--`

   (D) `int k = j; k >= 0; k--`

8. Consider the following code segment.

```
int j = 1;
while (j < 4)
{
   for (int k = 0; k <= 4; k++)
   {
      System.out.println("hello");   // line 6
   }
   j++;
}
```

   How many times will the statement in line 6 be executed as a result of running this code segment?

   (A) 12

   (B) 15

   (C) 16

   (D) 20

9. A programmer is creating a `Movie` class that will contain attributes for a movie's title and its average user rating on a scale from `0.0` to `5.0`. The class must also contain methods to allow the attributes to be accessed outside the class.

   A class design diagram for the `Movie` class will contain three sections. The first section contains the class name, the second section contains two instance variables and their data types, and the third section contains two methods and their return types. The `+` symbol indicates a `public` designation, and the symbol `-` indicates a `private` designation.

   Which of the following diagrams represents the most appropriate design for the `Movie` class?

(A)

Movie
+ title : String + rating : double
- getTitle() : String - getRating() : double

(B)

Movie
+ title : String + rating : int
- getTitle() : String - getRating() : int

(C)

Movie
- title : String - rating : double
+ getTitle() : String + getRating() : double

(D)

Movie
- title : String - rating : int
+ getTitle() : String + getRating() : int

10. A programmer developed a method and wants to allow other programmers to use the method in other programs without raising intellectual property concerns. Which of the following actions is most likely to support this goal?

(A) Designating the method as `public` in the method header

(B) Designating the method as `static` in the method header

(C) Publishing the code for the method as open source

(D) Removing all personal information from the data used by the method

11. The following class declarations are used to represent information about an apartment and a duplex, which is a building containing two apartments.

```
public class Apartment
{
   public int calculateRent()
   { /* implementation not shown */ }

   private String getTenant()
   { /* implementation not shown */ }

   /* There may be instance variables, methods,
      and constructors that are not shown. */
}

public class Duplex
{
   private Apartment unitOne;
   private Apartment unitTwo;

   public void printInfo()
   {
      System.out.println( /* missing code */ );
   }

   /* There may be instance variables, methods,
      and constructors that are not shown. */
}
```

Which of the following replacements for `/* missing code */` will compile without error?

(A) `Apartment.calculateRent()`

(B) `Apartment.getTenant()`

(C) `unitOne.calculateRent()`

(D) `unitTwo.getTenant()`

12. Consider the following class declaration.

```
public class Point
{
   private int xCoor;  // the x-coordinate of the point
   private int yCoor;  // the y-coordinate of the point

   public Point(int x, int y)
   {
      xCoor = x;
      yCoor = y;
   }

   public int getXCoor()
   {
      return xCoor;
   }

   public int getYCoor()
   {
      return yCoor;
   }
   public boolean isEquiv(Point otherP)
   {
      return /* missing expression */;
   }
}
```

Two `Point` objects are considered equivalent if they have the same *x* and *y* coordinates. The `isEquiv` method is intended to return `true` if this `Point` object is equivalent to the `Point` object parameter and return `false` otherwise.

Which of the following expressions can replace `/* missing expression */` so that the method works as intended?

(A) `getXCoor() == xCoor && getYCoor() == yCoor`

(B) `otherP.xCoor == x && otherP.yCoor == y`

(C) `otherP.getXCoor() == getXCoor() &&`
`otherP.getYCoor() == getYCoor()`

(D) `otherP == this`

13. A researcher wants to investigate which age group (under 18 years old, or 18 years old and over) is more likely to have downloaded a certain application from an online store. Each customer of the store has a unique ID number. The researcher has the following data sets available.

- Data set 1 contains an entry for each application available from the store. Each entry includes the name of the application and the total number of customers who have downloaded it.
- Data set 2 contains an entry for each application available from the store. Each entry includes the ID number and age of the customer who most recently downloaded the application.
- Data set 3 contains an entry for each customer of the store. Each entry includes the customer's ID number and the customer's age.
- Data set 4 contains an entry for each customer of the store. Each entry includes the customer's ID number and the names of all the applications downloaded by the customer.

Which two data sets can be combined and analyzed to determine the desired information?

(A) Data sets 1 and 2

(B) Data sets 2 and 3

(C) Data sets 2 and 4

(D) Data sets 3 and 4

14. Consider the following method, which is intended to return the maximum integer value contained in `ranges`.

```
/** Precondition: ranges contains at least one element. */
public int findMaxValue(int[] ranges)
{
   int max = 0;                  // line 4
   for (int value : ranges)
   {
      if (value > max)          // line 7
      {
         max = value;           // line 9
      }
   }
   return max;
}
```

This method does not always work as intended. Which of the following changes can be made so that the code segment works as intended?

(A) Changing line 4 to `int max = ranges[0];`

(B) Changing line 4 to `int max = Integer.MAX_VALUE;`

(C) Changing line 7 to `if (value < max)`

(D) Changing line 9 to `value = max;`

15. A text file named `roster.txt` has the following contents. Assume that the file contains no spaces.

```
Cohen-Isabel
Lee-Cindy
Patel-Anisha
Sanchez-Michael
```

In the following code segment, a valid `Scanner` object named `scan` has been created to read from the text file.

```
File myText = new File("roster.txt");
Scanner scan = new Scanner(myText);

ArrayList<String> usernames = new ArrayList<String>();

/* missing code */
```

This code segment is intended to assign `["ICohen", "CLee", "APatel", "MSanchez"]` to `usernames`.

Which of the following can replace `/* missing code */` so that this code segment works as intended?

(A)
```
while (scan.hasNext())
{
   String temp = scan.next().substring(0, 1);
   usernames.add(temp + scan.next());
}
```

(B)
```
while (scan.hasNext())
{
   String one = scan.next();
   String two = scan.next();
   usernames.add(two.substring(0, 1) + one);
}
```

(C)
```
while (scan.hasNext())
{
   String[] temp = scan.next().split("-");
   usernames.add(temp[1].substring(0, 1) + temp[0]);
}
```

(D)
```
while (scan.hasNext())
{
   String[] temp = scan.next().split("-");
   usernames.add(temp[0].substring(0, 1) + temp[1]);
}
```

16. Consider the following method.

```
public void reviseList(ArrayList<Integer> list)
{
   for (int i = 0; i < list.size(); i++)
   {
      int temp = list.get(i);
      if (temp % 2 == 1)
      {
         list.set(i, temp + 1);
      }
   }
}
```

The `reviseList` method is called with an `ArrayList` containing the values `[1, 2, 3, 3, 2, 1]`. What will be the contents of the `ArrayList` after executing this method?

(A) `[1, 3, 3, 3, 3, 1]`

(B) `[1, 3, 3, 4, 2, 2]`

(C) `[2, 2, 4, 4, 2, 2]`

(D) `[2, 3, 4, 4, 3, 2]`

17. Consider the following code segment.

```
int[][] arr2D = /* missing initialization */;
int sum = 0;

for (int j = 0; j < arr2D.length; j++)
{
   sum += arr2D[j][2];
}

System.out.println(sum);
```

Which of the following replacements for `/* missing initialization */` will cause the code segment to print the value `3`?

(A)
```
{{0, 0, 1},
 {0, 0, 2},
 {0, 0, 1}}
```

(B)
```
{{0, 0, 0},
 {0, 1, 2},
 {0, 0, 0}}
```

(C)
```
{{0, 0, 0},
 {0, 0, 0},
 {1, 1, 1},
 {0, 0, 0}}
```

(D)
```
{{0, 0, 1, 0},
 {0, 0, 1, 0},
 {0, 0, 1, 0}}
```

18. Consider the following code segment.

```
int[][] arr = {{10, -7, 19, 14},
               {-3, -2, -6, 11},
               {-9, 12, 10, -1}};
int ans = 1;
for (int[] row : arr)
{
   for (int value : row)
   {
      if (value > 0)
      {
         ans = ans * value;
      }
   }
}
System.out.println(ans);
```

Which of the following best describes the behavior of the code segment?

(A) It prints the product of all the elements in `arr`.

(B) It prints the product of all the negative elements in `arr`.

(C) It prints the product of all the positive elements in `arr`.

(D) It prints nothing because of a run-time error.

19. The following method is a correct implementation of the selection sort algorithm. The method sorts the elements of `arr` so that they are in order from least to greatest.

```
public static void selectionSort(int[] arr)
{
   for (int j = 0; j < arr.length - 1; j++)
   {
      int minIndex = j;
      for (int k = j + 1; k < arr.length; k++)
      {
         if (arr[k] < arr[minIndex])
         {
            minIndex = k;
         }
      }

      int temp = arr[j];
      arr[j] = arr[minIndex];
      arr[minIndex] = temp;
      /* end of outer loop */
   }
}
```

Assume that `selectionSort` has been called with an `int[]` argument that has been initialized with the following contents.

`{40, 30, 50, 60, 10, 20}`

What will the contents of `arr` be after three iterations of the outer loop (i.e., when `j == 2` at the point indicated by `/* end of outer loop */`)?

(A) `{10, 20, 30, 40, 60, 50}`

(B) `{10, 20, 30, 60, 40, 50}`

(C) `{10, 20, 50, 60, 40, 30}`

(D) `{10, 30, 50, 60, 40, 20}`

20. Consider the following method.

```
public static void mystery(int j, int k)
{
   if (j > k)
   {
      mystery(j – 2, k + 2);
   }
   System.out.print(j + " ");
}
```

What is printed as a result of the call `mystery(10, 0)` ?

(A) `10 8 6 4`

(B) `10 8 6`

(C) `6 8 10`

(D) `4 6 8 10`

Section II: Free-Response

The following are examples of the kinds of free-response questions found on the exam.

Question 1: Methods and Control Structures

This question involves the `MessageBuilder` class, which is used to generate a message based on a starting word. The `MessageBuilder` class contains a helper method, `getNextWord`. You will write a constructor and a method in the `MessageBuilder` class.

```
public class MessageBuilder
{
   private String message; // To be initialized in part (a)
   private int numWords;   // To be initialized in part (a)

   /**
    * Builds a message starting with the word specified by the
    * parameter and counts the number of words in the message,
    * as described in part (a)
    * Precondition: startingWord is a single word with no spaces.
    */
   public MessageBuilder(String startingWord)
   { /* to be implemented in part (a) */ }

   /**
    * Returns a word to follow the word specified by the
    * parameter or null if there are no remaining words.
    * Precondition: s is a single word with no spaces.
    * Postcondition: Returns an individual word with no spaces.
    */
   public String getNextWord(String s)
   { /* implementation not shown */ }

   /**
    * Returns an abbreviation for the instance variable message,
    * as described in part (b)
    * Preconditions: Each word in message is separated by a
    *                single space.
    *                message contains two or more words.
    * Postcondition: message is unchanged.
    */
   public String getAbbreviation()
   { /* to be implemented in part (b) */ }

   /* There may be instance variables, constructors, and methods
      that are not shown. */
}
```

(a) Write the `MessageBuilder` constructor, which uses a helper method to create a message starting with the word specified by the parameter `startingWord` and assigns the message to the instance variable `message`. The constructor also counts the number of words in the message and assigns the count to the instance variable `numWords`.

Each word in the message will be separated by a single space.

A helper method, `getNextWord`, has been provided to obtain words to be added to the message. Each call to `getNextWord` returns the next word in the message based on the word most recently added to the message. When there are no more words to be added to the message, `getNextWord` returns `null`. Consecutive calls to `getNextWord` are guaranteed to eventually return `null`.

Example 1

Consider the following calls to `getNextWord`, which are made within the `MessageBuilder` class. The return value of each call is used in the next call.

Method Call	Return Value
`getNextWord("the")`	`"book"`
`getNextWord("book")`	`"on"`
`getNextWord("on")`	`"the"`
`getNextWord("the")`	`"table"`
`getNextWord("table")`	`null`

Based on these return values, a call to the `MessageBuilder` constructor with argument `"the"` should set the instance variable message to `"the book on the table"` and should set the instance variable `numWords` to `5`.

Example 2

Consider the following calls to `getNextWord`, which are made within the `MessageBuilder` class. The return value of each call is used in the next call.

Method Call	Return Value
`getNextWord("good")`	`"morning"`
`getNextWord("morning")`	`"sunshine"`
`getNextWord("sunshine")`	`null`

Based on these return values, a call to the `MessageBuilder` constructor with argument `"good"` should set the instance variable `message` to `"good morning sunshine"` and should set the instance variable `numWords` to `3`.

Complete the `MessageBuilder` constructor. You must use `getNextWord` appropriately in order to receive full credit.

```
/**
 * Builds a message starting with the word specified by the
 * parameter and counts the number of words in the message,
 * as described in part (a)
 * Precondition: startingWord is a single word with no spaces.
 */
public MessageBuilder(String startingWord)
```

(b) Write the `getAbbreviation` method, which returns an abbreviation consisting of the first letter of each word in `message`. Assume that `message` consists of two or more words, each separated by a single space.

For example, if the value of `message` is `"as soon as possible"`, then `getAbbreviation()` should return `"asap"`.

Complete the `getAbbreviation` method.

```
/**
 * Returns an abbreviation for the instance variable message,
 * as described in part (b)
 * Preconditions: Each word in message is separated by a
 *                single space.
 *                message contains two or more words.
 * Postcondition: message is unchanged.
 */
public String getAbbreviation()
```

Question 2: Class Design

The `CupcakeMachine` class, which you will write, represents a cupcake vending machine, an automated machine that dispenses cupcakes. `CupcakeMachine` objects are created by calls to a constructor with two parameters.

- The first parameter is an `int` that represents the number of cupcakes that the vending machine has been stocked with. Assume that this value will be greater than or equal to `0`.
- The second parameter is a `double` that represents the cost, in dollars, per cupcake. Assume that this value will be greater than `0.0`.

The `CupcakeMachine` class contains a `takeOrder` method, which determines whether a cupcake order can be filled. A cupcake order can be filled if there are at least as many cupcakes in the vending machine as there are in the order.

A cupcake order is represented by a single `int` parameter to the `takeOrder` method. Assume that all values passed to the `takeOrder` method are positive.

If an order can be filled, the method updates the number of cupcakes remaining in the machine and returns a `String` containing information about the order. The returned string indicates the order number and the cost of the order, as shown in the following table. Order numbers begin at `1` and increase by `1` for each order filled.

If the order cannot be filled because the vending machine does not have enough cupcakes, the `takeOrder` method should return the message `"Order cannot be filled"`. In this case, the number of cupcakes available in the machine is unchanged and no order number is given to the order.

The following table contains a sample code execution sequence and the corresponding results. The code execution sequence appears in a class other than `CupcakeMachine`.

Statement	Return Value (blank if no value)	Explanation
`String info;`		
`CupcakeMachine c1 = new CupcakeMachine(10, 1.75);`		`CupcakeMachine c1` is constructed with 10 cupcakes and a cost of $1.75 per cupcake.
`info = c1.takeOrder(2);`	`"Order number 1, cost $3.5"`	A customer orders 2 cupcakes for a total cost of 2 × $1.75. There are now 8 cupcakes remaining in the machine.
`info = c1.takeOrder(3);`	`"Order number 2, cost $5.25"`	A customer orders 3 cupcakes for a total cost of 3 × $1.75. There are now 5 cupcakes remaining in the machine.
`info = c1.takeOrder(10);`	`"Order cannot be filled"`	A customer attempts to order 10 cupcakes. The machine only has 5 cupcakes available, so no order is made.
`info = c1.takeOrder(1);`	`"Order number 3, cost $1.75"`	A customer orders 1 cupcake for a total cost of $1.75. There are now 4 cupcakes remaining in the machine.
`CupcakeMachine c2 = new CupcakeMachine(10, 1.5);`		`CupcakeMachine c2` is constructed with 10 cupcakes and a cost of $1.50 per cupcake.
`info = c2.takeOrder(10);`	`"Order number 1, cost $15.0"`	A customer orders 10 cupcakes for a total cost of 10 × $1.50. There are now 0 cupcakes remaining in the machine.

Write the complete `CupcakeMachine` class. Your implementation must meet all specifications and conform to the examples shown in the table.

Question 3: Data Analysis with `ArrayList`

The `ItemInfo` class is used to store information about an item at a store. A partial declaration of the `ItemInfo` class is shown.

```
public class ItemInfo
{
   /**
    * Returns the name of the item
    */
   public String getName()
   { /* implementation not shown */ }

   /**
    * Returns a value greater than 0.0 that represents the
    * cost of a single unit of the item, in dollars
    */
   public double getCost()
   { /* implementation not shown */ }

   /**
    * Returns true if the item is currently available and
    * returns false otherwise
    */
   public boolean isAvailable()
   { /* implementation not shown */ }

   /* There may be instance variables, constructors, and
      methods that are not shown. */
}
```

The `ItemInventory` class maintains an `ArrayList` named `inventory` that contains all items at the store. A partial declaration of the `ItemInventory` class is shown.

```
public class ItemInventory
{
   /** The list of all items at the store */
   private ArrayList<ItemInfo> inventory;

   /**
    * Returns the average cost of the available items
    * whose cost is between lower and upper, inclusive
    * Precondition: lower <= upper
    *               At least one available element of
    *               inventory has a cost between
    *               lower and upper, inclusive.
    *               No elements of inventory are null.
    */
   public double averageWithinRange(double lower, double upper)
   { /* to be implemented */}

   /* There may be instance variables, constructors, and methods
      that are not shown. */
}
```

Write the `ItemInventory` method `averageWithinRange`. The method should return the average cost of the available items in `inventory` whose cost is between the parameters `lower` and `upper`, inclusive.

Suppose `inventory` contains the following seven `ItemInfo` objects.

Name	"action figure"	"hair brush"	"frying pan"	"dish sponge"	"coffee mug"	"scarf"	"watch"
Cost	20.0	7.99	45.0	2.0	10.0	59.0	45.0
Is Available	true	true	true	false	true	true	false

For the inventory shown, `averageWithinRange(10.0, 50.0)` should return `25.0`, which is equal to the average cost of the available items within the specified range (a $20 action figure, a $45 frying pan, and a $10 coffee mug). Although the watch is within the specified range, it is not available.

Complete method `averageWithinRange`.

```
/**
 * Returns the average cost of the available items
 * whose cost is between lower and upper, inclusive
 * Precondition: lower <= upper
 *               At least one available element of
 *               inventory has a cost between
 *               lower and upper, inclusive.
 *               No elements of inventory are null.
 */
public double averageWithinRange(double lower, double upper)
```

Question 4: 2D Array

The `Appointment` class is used to store information about a person's scheduled appointments. A partial declaration of the `Appointment` class is shown.

```
public class Appointment
{
   /**
    * Returns the status of the appointment ("free", "busy", etc.)
    */
   public String getStatus()
   { /* implementation not shown */ }

   /**
    * Returns the room number of the appointment location
    */
   public int getRoomNumber()
   { /* implementation not shown */ }

   /* There may be instance variables, constructors, and methods
      that are not shown. */
}
```

The `Schedule` class maintains a two-dimensional array of appointments that represents a person's schedule. A partial declaration of the `Schedule` class is shown.

```
public class Schedule
{
   private Appointment[][] sched;

   /**
    * Returns the index of a column containing the fewest
    * occurrences of the status indicated by the parameter
    * target
    * Preconditions: sched is not null and no elements
    *                of sched are null.
    *                sched has at least one row and at
    *                least one column.
    */
   public int columnWithFewest(String target)
   { /* to be implemented */}

   /* There may be instance variables, constructors, and methods
      that are not shown. */
}
```

When an element of the two-dimensional array `sched` is accessed, the first index is used to specify the row and the second index is used to specify the column.

Write the `Schedule` method `columnWithFewest`. The method should return the index of a column in `sched` that contains the fewest occurrences of the parameter `target`. If there are multiple columns that have the fewest number of occurrences of `target`, any of their column indices can be returned.

Suppose `sched` has the following contents. For each element, the first value is the appointment status and the second value is the room number.

	0	1	2	3	4
0	"free" 100	"free" 100	"free" 100	"busy" 206	"busy" 204
1	"free" 100	"free" 100	"busy" 304	"busy" 206	"busy" 202
2	"hold" 201	"busy" 105	"busy" 205	"free" 100	"busy" 205
3	"busy" 204	"free" 100	"busy" 310	"hold" 110	"free" 100
4	"busy" 204	"hold" 201	"hold" 310	"busy" 105	"free" 100
5	"busy" 105	"busy" 208	"hold" 310	"busy" 105	"free" 100

For these contents of `sched`, the expected behavior of `columnWithFewest` is as follows.

- The call `columnWithFewest("busy")` should return `1` because `"busy"` appears two times in column `1` and more than two times in each of the other columns.
- The call `columnWithFewest("free")` should return either `2` or `3` because `("free")` appears one time in column `2`, one time in column `3`, and more than one time in each of the other columns.
- The call `columnWithFewest("hold")` should return `4` because `"hold"` appears zero times in column `4` and one or more times in each of the other columns.

Complete method `columnWithFewest`.

```
/**
 * Returns the index of a column containing the fewest
 * occurrences of the status indicated by the parameter
 * target
 * Preconditions: sched is not null and no elements
 *                of sched are null.
 *                sched has at least one row and at
 *                least one column.
 */
public int columnWithFewest(String target)
```