

# COMPUTER SCIENCE PRINCIPLES

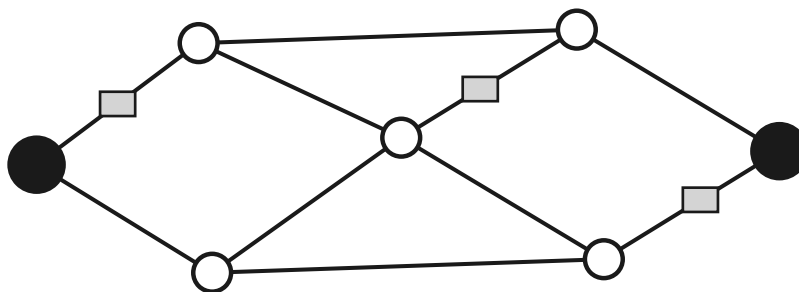
## REVIEW WORKBOOK

---

**2027 EXAM**

Algorithms • Data • Networks • Create Task Prep

*Think computationally. Explain clearly. Build with purpose.*



01000001 01010000 00101011

Updated for the May 2027 digital exam • Diagnostic test • 5 Big Ideas • Full-length practice exam

**THIS BOOK BELONGS TO**

Name \_\_\_\_\_

School / Teacher \_\_\_\_\_

AP exam date \_\_\_\_\_

# Contents

## GETTING STARTED

|                                                    |   |
|----------------------------------------------------|---|
| How to Use This Workbook .....                     | 4 |
| The 2027 AP Computer Science Principles Exam ..... | 5 |
| 8-Week Study Plan and Progress Tracker .....       | 6 |
| Reading AP Pseudocode .....                        | 8 |
| Diagnostic Test .....                              | 9 |

## BIG IDEAS

|                                               |    |
|-----------------------------------------------|----|
| <b>1</b> Creative Development .....           | 25 |
| <b>2</b> Data .....                           | 38 |
| <b>3</b> Algorithms and Programming .....     | 54 |
| <b>4</b> Computing Systems and Networks ..... | 78 |
| <b>5</b> Impact of Computing .....            | 90 |

## CREATE TASK AND EXAM PREP

|                                             |     |
|---------------------------------------------|-----|
| Code Tracing and Pseudocode Lab .....       | 106 |
| Create Task Planning Lab .....              | 110 |
| Written Response Builder .....              | 114 |
| Internet and Cybersecurity Review Lab ..... | 118 |
| Common Mistake Fixer .....                  | 121 |
| Full-Length 2027 Practice Exam .....        | 123 |

## ANSWERS

|                                    |     |
|------------------------------------|-----|
| Answer Key .....                   | 148 |
| Final Tracker and Next Steps ..... | 157 |

### **i** KNOW IT

Short concept notes in the margin, next to the problems that use them.

### **WR** EDGE

Written-response practice with a labeled box for each prompt.

### **!** FIX IT

The mistakes students make most often, named so you can catch them.

## 1

**AP EXAM WEIGHT: 10–13% OF THE MULTIPLE-CHOICE SECTION**

## How do people turn an idea into a program that works for the people who use it, and how do they know it works?

- ☐ Explain how collaboration and diverse perspectives improve a program.
- ☐ Distinguish a program's purpose from its function, and identify inputs, outputs, and events.
- ☐ Compare iterative and incremental development, and plan a program from user needs.
- ☐ Document code and acknowledge code written by others.
- ☐ Classify syntax, run-time, logic, and overflow errors, and choose tests that find them.

- 1.1** Collaboration
- 1.2** Program Function and Purpose
- 1.3** Program Design and Development
- 1.4** Identifying and Correcting Errors

Unit Review: Mixed Practice, WR Edge, Fix It, Exit Ticket, READY+ Check

Answer in a few words each. These ideas run through the whole unit.

|   |   |   |   |   |   |   |   |   |    |
|---|---|---|---|---|---|---|---|---|----|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

[illegible]

A rectangular box containing a 4x10 grid of dots, intended for drawing a picture.

## UNIT 1 | LESSON 1.1

# Collaboration

## i KNOW IT

Programs are better when people with **different perspectives** help build them. Different backgrounds catch different problems and reduce bias in the design. Common practices: **pair programming**, code review, shared planning documents, and online tools that let a team edit and comment together.

## CED TOPIC

### 1.1 Collaboration

## ! FIX IT

Treating collaboration as copying. Collaborators divide work, discuss design choices, and review each other's code. Each person must still understand the whole program.

## i WORKED EXAMPLE: WHAT COLLABORATION ADDS

A team of three is building an app that helps students track volunteer hours. One member has a part-time job, one uses a screen reader, and one has never volunteered before.

**PERSPECTIVE 1** The working student asks for a way to log hours from a phone in under a minute.

**PERSPECTIVE 2** The screen-reader user points out that every button needs a text label.

**PERSPECTIVE 3** The newcomer asks what counts as volunteering, so the team adds examples to the help screen.

**RESULT** Three improvements that no single member would have thought of alone.

## 🕒 TRY IT

**CONCEPT**

- 1 During pair programming, the **driver** types and the **navigator** reviews. Describe one thing the navigator should watch for, and why it helps.

---



---



---

- 2 A team building a bus-schedule app includes only students who drive to school. Explain one problem the app might have because of who was on the team.

---



---



---

## 🕒 PRACTICE: GIVING USEFUL FEEDBACK

**CONCEPT**

A teammate's program crashes whenever a user types a word where a number is expected. Feedback is most useful when it is specific, explains the effect, and suggests a next step.

- 3 Rewrite this weak feedback so it is specific and useful: "Your input part is broken."

---



---



---

- 4 Name two tools or practices a remote team could use to collaborate on the same program, and what each is for.

---



---

## UNIT 1 | LESSON 1.2

# Program Function and Purpose

## i WORKED EXAMPLE: DESCRIBING A PROGRAM

A **parking-spot finder** shows nearby open spots after the driver taps “Find parking.”

| PART     | DESCRIPTION                                                                             |
|----------|-----------------------------------------------------------------------------------------|
| Purpose  | Help drivers park quickly without circling the block.                                   |
| Function | Reads the phone’s location, requests open-spot data, and lists the three closest spots. |
| Inputs   | The tap, the phone’s location, and the spot data from the city’s server.                |
| Output   | A list of spots with walking distances.                                                 |
| Event    | Tapping “Find parking” runs the search code.                                            |

## © TRY IT

### CONCEPT

- 1 A **plant-watering reminder** app sends an alert when a soil sensor reports dry soil. State its purpose, one input, one output, and the event that triggers the alert.

---



---



---



---

- 2 Classify each statement about a **homework planner** app as purpose or function.

| STATEMENT                                                    | PURPOSE OR FUNCTION? |
|--------------------------------------------------------------|----------------------|
| Helps students avoid missing due dates.                      |                      |
| Sorts assignments by due date and highlights the next three. |                      |
| Sends a notification at 7 p.m. the day before each due date. |                      |
| Reduces stress during exam weeks.                            |                      |

- 3 Explain why the same program can behave differently each time it runs, even though its code never changes. Use the word **event** in your answer.

---



---



---



---



---



---



---



---



---



---

## i KNOW IT

**Purpose:** the problem a program solves or the need it meets. Why it exists.

**Function:** what it actually does as it runs. How it behaves.

**Input:** data sent to a program, from a user, a sensor, a file, or another program.

**Output:** data a program sends out, such as text, a sound, or a file.

## EVENTS

An **event** is an action, such as a click or a key press, that triggers code. Programs that respond to events are **event-driven**: their order depends on what the user does.

## CED TOPIC

1.2 Program Function and Purpose

## Testing a Discount Program

A school store gives a 10 percent discount on any order of 20 dollars or more. The procedure below is supposed to return the amount a customer pays.

```
PROCEDURE amountDue(total)
{
    IF(total > 20)
    {
        RETURN(total - total / 10)
    }
    RETURN(total)
}
```

A customer reports that an order of exactly 20 dollars was charged the full price. Answer each prompt with specific values.

**Program function and purpose.** Describe the purpose of `amountDue` and what it does with its input.

**Errors and testing.** Identify the error. Give the input that reveals it, the expected result, and the actual result.

## WR EDGE • UNIT 1 • CONTINUED

**c Correcting errors.** Rewrite the one line that must change, and name two more test inputs you would use to confirm the fix, with their expected results.

**d Documentation.** Write a comment for `amountDue` that would help another programmer use it correctly, and explain why documentation matters when a team shares code.



**✓ READY+ CHECK • SCORE YOURSELF**

**CONFIDENCE** (1) (2) (3) (4)

- ☐ I gave exact inputs and results, not general descriptions.
- ☐ My tests included the boundary value.

- ☐ I named the kind of error correctly.
- ☐ My comment states what the procedure returns and when.

UNIT 1 | REVIEW

Fix It and Exit Ticket

! FIX IT: COMMON MISTAKES IN UNIT 1

| MISTAKE                                                   | FIX                                                                                                                             |
|-----------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------|
| Describing what the code does when asked for its purpose. | Purpose is the <b>why</b> : the problem solved or the need met. Function is the <b>what</b> .                                   |
| Testing only typical values.                              | Test values at, just below, and just above every boundary, plus unusual inputs.                                                 |
| Calling every wrong answer a syntax error.                | Syntax: will not run. Run-time: stops while running. Logic: runs with the wrong result. Overflow: value too large for its bits. |
| Leaving borrowed or AI-generated code uncredited.         | Acknowledge the source in a comment, and make sure you can explain every line.                                                  |
| Mixing up iterative and incremental.                      | Iterative repeats cycles with feedback. Incremental adds and tests one piece at a time.                                         |

Which of these mistakes are you most likely to make, and how will you catch it?

→ EXIT TICKET

Complete without looking back.

1 A smart thermostat turns on the heat when a person taps “Warm up.” Name the event, one input, and one output.

2 A procedure should return true for every test score from 70 through 100. List three test values you would use and why.

3 In one sentence each, define a logic error and a run-time error.

## Section I Answer Sheet

Fill in one bubble per question, or two bubbles for each **SELECT TWO** question (marked with a square below).

### SECTION I • QUESTIONS 1 TO 70

⌚ 120 MINUTES

- 1     (A) (B) (C) (D)
- 4     (A) (B) (C) (D)
- 7     (A) (B) (C) (D)
- 10    (A) (B) (C) (D)
- 13    (A) (B) (C) (D)
- 16    (A) (B) (C) (D)
- 19    (A) (B) (C) (D)
- 22    (A) (B) (C) (D)
- 25    (A) (B) (C) (D)
- 28    (A) (B) (C) (D)
- 31    (A) (B) (C) (D)
- 34    (A) (B) (C) (D)
- 37    (A) (B) (C) (D)
- 40    (A) (B) (C) (D)
- 43    (A) (B) (C) (D)
- 46    (A) (B) (C) (D)
- 49    (A) (B) (C) (D)
- 52    (A) (B) (C) (D)
- 55    (A) (B) (C) (D)
- 58    (A) (B) (C) (D)
- 61    (A) (B) (C) (D)
- 64    (A) (B) (C) (D)
- 67 ■ (A) (B) (C) (D)
- 70    (A) (B) (C) (D)

- 2     (A) (B) (C) (D)
- 5     (A) (B) (C) (D)
- 8     (A) (B) (C) (D)
- 11    (A) (B) (C) (D)
- 14 ■ (A) (B) (C) (D)
- 17    (A) (B) (C) (D)
- 20    (A) (B) (C) (D)
- 23    (A) (B) (C) (D)
- 26    (A) (B) (C) (D)
- 29    (A) (B) (C) (D)
- 32    (A) (B) (C) (D)
- 35    (A) (B) (C) (D)
- 38    (A) (B) (C) (D)
- 41    (A) (B) (C) (D)
- 44    (A) (B) (C) (D)
- 47    (A) (B) (C) (D)
- 50    (A) (B) (C) (D)
- 53    (A) (B) (C) (D)
- 56    (A) (B) (C) (D)
- 59 ■ (A) (B) (C) (D)
- 62    (A) (B) (C) (D)
- 65    (A) (B) (C) (D)
- 68    (A) (B) (C) (D)

- 3     (A) (B) (C) (D)
- 6     (A) (B) (C) (D)
- 9     (A) (B) (C) (D)
- 12    (A) (B) (C) (D)
- 15    (A) (B) (C) (D)
- 18    (A) (B) (C) (D)
- 21 ■ (A) (B) (C) (D)
- 24    (A) (B) (C) (D)
- 27 ■ (A) (B) (C) (D)
- 30    (A) (B) (C) (D)
- 33    (A) (B) (C) (D)
- 36    (A) (B) (C) (D)
- 39    (A) (B) (C) (D)
- 42    (A) (B) (C) (D)
- 45    (A) (B) (C) (D)
- 48 ■ (A) (B) (C) (D)
- 51    (A) (B) (C) (D)
- 54    (A) (B) (C) (D)
- 57    (A) (B) (C) (D)
- 60 ■ (A) (B) (C) (D)
- 63    (A) (B) (C) (D)
- 66    (A) (B) (C) (D)
- 69 ■ (A) (B) (C) (D)

Total correct: \_\_\_\_\_ / 70

**Select-two items:** both answers must be correct to earn the point.

# Answer Key

Answers are grouped by section, with the page number of each set of problems. Multiple-choice answers include the reason the answer is correct. Written-response answers show one strong response; other wording earns credit when it makes the same points with the same specific details. Check a whole section at a time, and redo anything you missed before reading the explanation.

## GETTING STARTED

### Reading AP Pseudocode • Self-Check p. 8

- 1 3
- 2 2
- 3 false

## DIAGNOSTIC TEST

### Diagnostic • Part B p. 20

- a Input: the word the student types (passed in as word), or the stored descriptions. Output: the list of descriptions that contain the word.
- b matches starts empty, and each description that contains the word is appended, so any number of matches can be collected and returned together as one value.
- c Iteration: FOR EACH d IN descriptions visits every stored description. Selection: IF (contains(d, word)) decides whether the current description is added to matches.
- d Example: findItems(["blue water bottle", "red scarf", "blue jacket"], "blue") should return ["blue water bottle", "blue jacket"]. It checks that every matching description, and only those, is returned. A second test with no matches should return an empty list.

## UNIT 1 • CREATIVE DEVELOPMENT

### Unit 1 Quick Start p. 26

- a A quantity and its unit, such as 250 grams.
- b The converted quantity, such as about 8.8 ounces.
- c Any typical value plus an unusual one, such as 0 or a very large amount, to see whether the conversion still works.

### Lesson 1.1 • Try It p. 26

- 1 Examples: logic errors, missed edge cases, or unclear names. A second reviewer catches mistakes as they are written, when they are cheapest to fix.
- 2 The team may not understand bus riders' needs, such as transfer times or late-bus alerts, so the app could leave out features its actual users need most.

### Lesson 1.1 • Feedback p. 26

- 3 Example: "When I type 'ten' for the age, the program stops with an error. Can we check that the input is a number and ask again if it is not?"
- 4 Examples: a shared code repository so everyone works from the latest version; comments or code review to suggest changes; a shared task list to divide work; video calls for pair programming.

### Lesson 1.2 • Try It p. 27

- 1 Purpose: keep plants healthy by reminding owners to water them. Input: soil-moisture readings. Output: the alert. Event: a reading below the dryness threshold.
- 2 Purpose, function, function, purpose.
- 3 An event-driven program runs code in response to events such as clicks and key presses. Users trigger different events in different orders, so the program's behavior depends on what each user does.

### Lesson 1.3 • Try It p. 28

- 1 (a) incremental; (b) iterative.
- 2 Examples: a survey, interviews, or watching people do the task now. Each reveals what users need and where they struggle.

### Lesson 1.3 • Documentation p. 28

- 3 Example: "Returns the average of the values in nums after dropping the single lowest value."
- 4 90. The total is 340, the lowest value is 70, and  $(340 - 70) / 3 = 90$ .
- 5 Example comment: "Sorting procedure generated with help from an AI chatbot on (date); modified to sort in descending order." The student is responsible for all submitted code and must be able to explain how it works.
- 6 Interview likely users, build a prototype, test it with users, revise the design. The build, test, and revise steps repeat in each iteration.
- 7 Each cycle produces something users can try, and their feedback shows which features matter before the team invests time in the wrong ones.

### Lesson 1.4 • Try It p. 30

- 1 Run-time error: index 4 is past the end of a 3-element list.
- 2 Overflow error: 260 does not fit in 8 bits (the largest value is 255).
- 3 Logic error: the program runs but gives the wrong result.
- 4 Syntax error.

### Lesson 1.4 • Testing p. 30

- 5 90: A, A. 85: B, B. 80: B, **C or lower**. 79: C or lower, C or lower.
- 6 The test with 80. Change `pct > 80` to `pct ≥ 80`.

### Unit 1 Review • Mixed Practice (Code) p. 31

- 1 **B**. After the first REMOVE, the list is ["dry", "fold"], which has only 2 elements. Asking to remove index 3 causes a run-time error.
- 2 **D**. 0 is not positive, but  $0 \geq 0$  is true, so the procedure returns true. The other calls return the intended values.
- 3 **A**. The running total is displayed after each addition: 4, then 11, then 12. Watching values change like this is a common debugging strategy.