AP COMPUTER SCIENCE PRINCIPLES • 2026-27

Algorithms before code.

Learn to describe a problem precisely, plan a solution visually, test it, and explain why it works.

Unit 1 · 7 lessons · 10 × 85-minute blocks

Student: _____ **Class:** _____

Essential questions

- What makes instructions precise enough for a computer?
- How do purpose, decomposition, and abstraction shape useful programs?
- How can we test a claim about a program?
- When is a good solution not necessarily the best solution?

By the end of this unit, I can...

- ☐ Design and trace an algorithm
- ☐ Explain program purpose, input, output, and behavior
- ☐ Construct a top-down diagram and reusable procedure
- ☐ Debug a mistake using evidence from test cases
- ☐ Compare simple algorithm-growth patterns

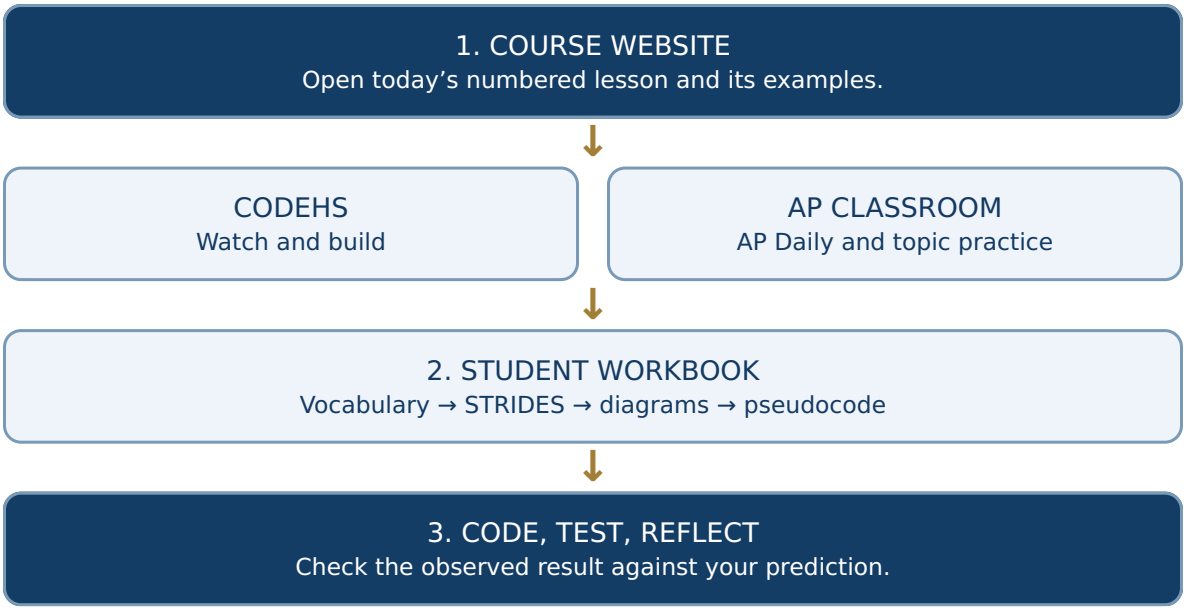
This is your engineering notebook

Complete it while using the **course website**, **CodeHS AP CSP in Python**, and **AP Classroom**. Your programming runs in CodeHS; your reasoning belongs here.

GUIDE

How the course resources fit together

Your website sets the order of instruction. CodeHS and AP Classroom supply selected activities from different sequences.

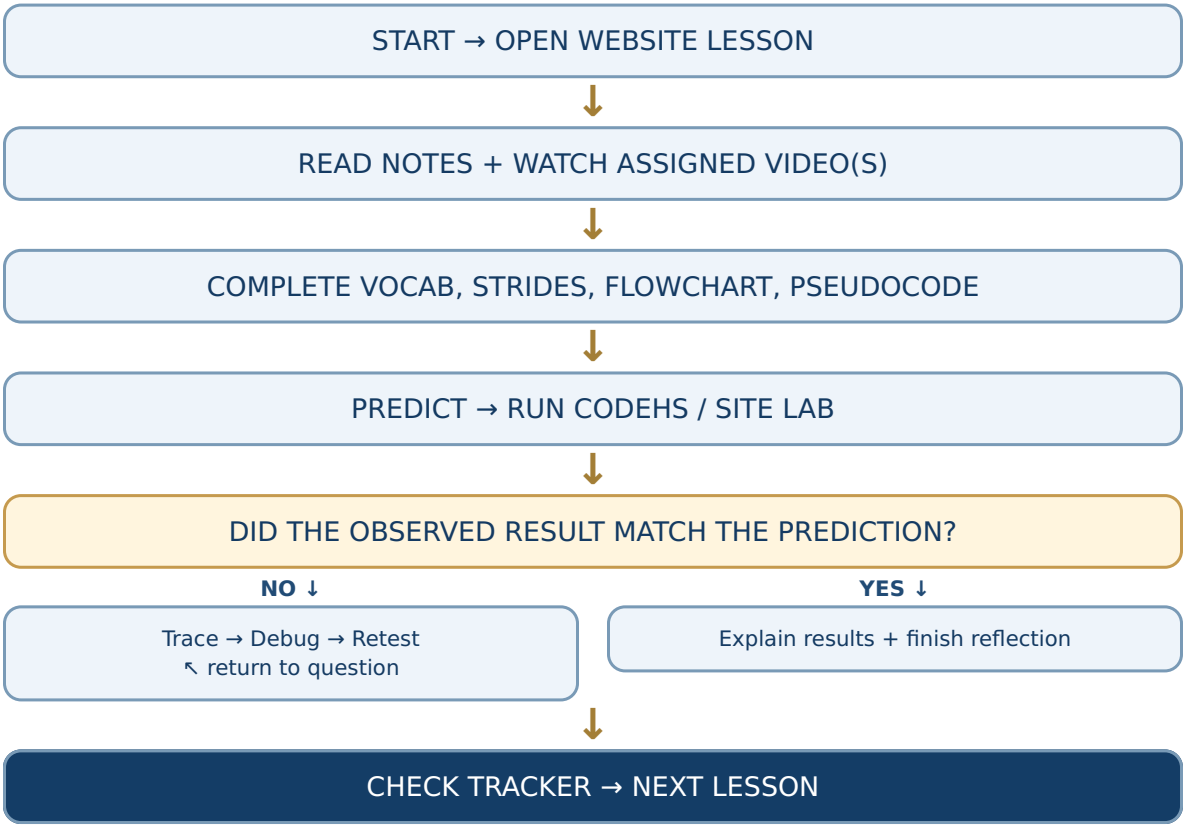


A **Where should you look first when CodeHS places a topic in a different order?**

B **Where should you record your prediction before running a program?**

GUIDE

Your repeatable lesson workflow



A **Which step should happen before you run the program?**

B **If a result is unexpected, should you guess, copy a solution, or trace your program? Why?**

GUIDE

STRIDES notes — a completed model

STRIDES helps your notes become a resource you can return to when solving a new problem.

S · Subject

Computer science and algorithm design

T · Topic

Precise instructions and starting conditions

R · Rules

A robot executes commands in sequence; MOVE changes its position; LEFT changes only its direction.

I · Illustrations

Start (1,1) facing east → MOVE → (2,1) east → LEFT → (2,1) north

D · Definitions

Algorithm: a finite, precise series of instructions.

E · Example

MOVE, LEFT, MOVE changes the robot from (1,1) east to (2,2) north (positive y upward).

S · Steps

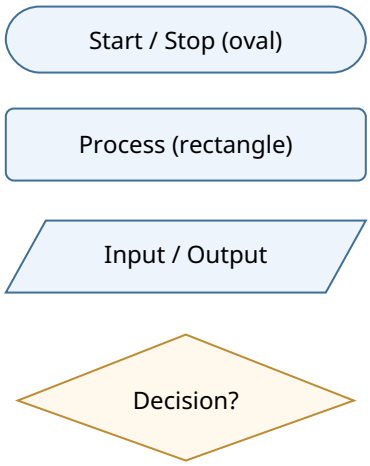
Identify the initial state → predict each command → record states → compare to goal.

A **Which STRIDES section is most useful for showing how a variable changes over time?**

GUIDE

Flowcharts and pseudocode — model then practice

Visual conventions



Sample problem

Ask for a temperature and display an alert when it is below 0°C.

```
START
INPUT temperature
IF temperature < 0
    DISPLAY "Freezing"
ELSE
    DISPLAY "Not freezing"
END IF
STOP
```

Important

Pseudocode communicates logic; it does not need perfect Python syntax. Later units will use the AP exam reference notation in more detail.

A **Draw the temperature-alert flowchart. Label the true and false paths.**

Begin with an input parallelogram. Add a decision diamond and two labeled output paths.

GUIDE

Predict → test → debug → reflect

A strong programming reflection cites evidence. A weak one only says “it worked.”

Prediction

A score of 70 should be accepted under the rule “70 or higher passes.”

Program condition

```
IF score > 70 THEN DISPLAY "PASS" ELSE DISPLAY "RETRY"
```

Input	Expected	Observed	Match?
69	RETRY	RETRY	Yes
70	PASS	RETRY	No
71	PASS	PASS	Yes

A useful reflection

“My test at 70 revealed a **logic error**. The condition excluded the boundary. I changed > to ≥ and retested 69, 70 and 71.”

A **What evidence would convince you the correction worked?**

B **Why is “it was easy” not a useful programming reflection?**

GUIDE

Assignment tracker — Unit 1

Check the specific activities listed on each lesson's first page. AP Classroom is accessed through your assigned class; if a topic video is not assigned, use this as a reference rather than guessing a private link.

Lesson	Block	AP topic(s)	Completed
1.1 Precise Algorithms	1-2	1.2 / 1.3; preview 3.9	☐ Web ☐ CodeHS ☐ AP ☐ Notes
1.2 Program Purpose & Iterative Design	3-4	1.1 / 1.2 / 1.3	☐ Web ☐ CodeHS ☐ AP ☐ Notes
1.3 Decomposition & Top-Down Design	5	1.3; preview 3.12	☐ Web ☐ CodeHS ☐ AP ☐ Notes
1.4 Abstraction & Reusable Procedures	6	1.3; preview 3.12 / 3.13	☐ Web ☐ CodeHS ☐ AP ☐ Notes
1.5 Errors, Debugging & Corrections	7	1.4	☐ Web ☐ CodeHS ☐ AP ☐ Notes
1.6 Testing, Tracing & Counterexamples	8-9	1.4; preview 3.5 / 3.9	☐ Web ☐ CodeHS ☐ AP ☐ Notes
1.7 Efficiency, Heuristics & Limits (Preview)	10 / extension	3.17 / 3.18 preview; revisit Unit 5	☐ Web ☐ CodeHS ☐ AP ☐ Notes

Lab evidence checklist

☐ Lab A: Precision Robot — predict, test, revise

☐ Lab B: Trace — states and displayed result

☐ Lab C: Debug — identify boundary error

☐ Lab D: Growth + routing heuristic — compare and explain

Unit 1 submission

Turn in this workbook as a unit classwork grade after completing the unit review. CodeHS assignments are completed and submitted inside CodeHS; do not copy source code into every reflection.

LESSON 1.1 • LEARN & RECORD

Precise Algorithms

I can: Write, trace, and revise a precise finite sequence of instructions.

Website: lesson 1.1
CodeHS: 1.1.1 Welcome to AP CSP; 1.2.1 Introduction to Programming With Karel; 1.2.4 Your First Karel Program; 1.3.4 Make a Tower
AP Classroom: 1.2 / 1.3; preview 3.9 (assigned AP Daily / topic questions)

Define the vocabulary in your own words

Term	Your definition + example
algorithm	
sequence	
precondition	
state	

Guided video and website notes

1 **What makes an instruction unambiguous?**

2 **Why does a starting direction matter?**

3 **What must be true for an algorithm to be correct?**

One rule and an illustration to remember

Add a rule, small diagram, or example that you can use on a later assignment.

LESSON 1.1 • VISUALIZE & PLAN

Precise Algorithms

Read the model. Predict or draw before looking at the worked solution on the website.

Modeled problem

A robot starts at (1,1) facing east. MOVE, MOVE, LEFT, MOVE. Positive y points north.

A **Describe what happens step by step. Label every important state or decision.**

Diagram your reasoning

Use boxes, arrows, and labels. Include the initial state, intermediate decisions, and the result.

B **Write the corresponding algorithm in readable pseudocode. Mark input, process and output when relevant.**

STRIDES check

Which illustration or rule from today would be useful when you encounter a similar task later?

LESSON 1.1 • APPLY & REFLECT

Precise Algorithms

Complete the assigned CodeHS activity and the matching website investigation. Predict before running.

Transfer challenge

Write a 4-command algorithm to move two cells east and one north. Start east at (1,1). Include a precondition.

Plan → test → revise

A **Write your initial algorithm or solution before using a computer.**

Test / starting state	Expected outcome	Observed outcome
_____	_____	_____
_____	_____	_____
_____	_____	_____

B **What did the first test reveal? Describe a change or explain why no revision was needed.**

C **Why can an algorithm be perfectly precise yet incorrect?**

☐ Website notes & checks ☐ CodeHS activities ☐ AP Daily/topic work ☐ Workbook

LESSON 1.2 • LEARN & RECORD

Program Purpose & Iterative Design

I can: Explain a program's purpose, behavior, inputs, outputs, and iterative improvement.

Website: lesson 1.2
CodeHS: 1.1.2 Why AP CSP?; 3.1.1 What is Code? (preview); 3.2.1 Uses of Programs (preview)
AP Classroom: 1.1 / 1.2 / 1.3 (assigned AP Daily / topic questions)

Define the vocabulary in your own words

Term	Your definition + example
program purpose	_____
program behavior	_____
input	_____
output	_____

Guided video and website notes

1 **How is purpose different from behavior?**

2 **What role does user feedback play?**

3 **What makes development iterative?**

One rule and an illustration to remember

Add a rule, small diagram, or example that you can use on a later assignment.

LESSON 1.2 • VISUALIZE & PLAN

Program Purpose & Iterative Design

Read the model. Predict or draw before looking at the worked solution on the website.

Modeled problem

School shuttle alert: input is minutes until arrival. When minutes ≤ 5 and ≥ 0, display “Get ready.”

A **Describe what happens step by step. Label every important state or decision.**

Diagram your reasoning

Use boxes, arrows, and labels. Include the initial state, intermediate decisions, and the result.

B **Write the corresponding algorithm in readable pseudocode. Mark input, process and output when relevant.**

STRIDES check

Which illustration or rule from today would be useful when you encounter a similar task later?

LESSON 1.2 • APPLY & REFLECT

Program Purpose & Iterative Design

Complete the assigned CodeHS activity and the matching website investigation. Predict before running.

Transfer challenge

Draw an input-process-output diagram for an app that reminds a student to bring a library book on its due date.

Plan → test → revise

A **Write your initial algorithm or solution before using a computer.**

Test / starting state	Expected outcome	Observed outcome
_____	_____	_____
_____	_____	_____
_____	_____	_____

B **What did the first test reveal? Describe a change or explain why no revision was needed.**

C **A program correctly performs its steps but fails its users. Explain.**

☐ Website notes & checks ☐ CodeHS activities ☐ AP Daily/topic work ☐ Workbook

LESSON 1.3 • LEARN & RECORD

Decomposition & Top-Down Design

I can: Break a complex problem into testable, meaningful subtasks.

Website: lesson 1.3
CodeHS: 1.5.1 Functions in Karel; 1.6.1 Top Down Design and Decomposition; 1.6.4 The Two Towers; 1.7.1 Commenting Your Code
AP Classroom: 1.3; preview 3.12 (assigned AP Daily / topic questions)

Define the vocabulary in your own words

Term	Your definition + example
decomposition	_____
top-down design	_____
subtask	_____
procedure	_____

Guided video and website notes

1 **What is an effective subtask boundary?**

2 **How does decomposition help testing?**

3 **Does decomposing always speed up a program?**

One rule and an illustration to remember

Add a rule, small diagram, or example that you can use on a later assignment.

LESSON 1.3 • VISUALIZE & PLAN

Decomposition & Top-Down Design

Read the model. Predict or draw before looking at the worked solution on the website.

Modeled problem

Delivery robot: DELIVER_PACKAGE includes NAVIGATE, CHECK_ROOM, DROP_PACKAGE, CONFIRM.

A **Describe what happens step by step. Label every important state or decision.**

Diagram your reasoning

Use boxes, arrows, and labels. Include the initial state, intermediate decisions, and the result.

B **Write the corresponding algorithm in readable pseudocode. Mark input, process and output when relevant.**

STRIDES check

Which illustration or rule from today would be useful when you encounter a similar task later?

LESSON 1.3 • APPLY & REFLECT

Decomposition & Top-Down Design

Complete the assigned CodeHS activity and the matching website investigation. Predict before running.

Transfer challenge

Decompose “publish a school club newsletter” into four meaningful subtasks, then subdivide one of them.

Plan → test → revise

A **Write your initial algorithm or solution before using a computer.**

Test / starting state	Expected outcome	Observed outcome
_____	_____	_____
_____	_____	_____
_____	_____	_____

B **What did the first test reveal? Describe a change or explain why no revision was needed.**

C **Why is “make one function per movement” often a weak decomposition?**

☐ Website notes & checks ☐ CodeHS activities ☐ AP Daily/topic work ☐ Workbook

LESSON 1.4 • LEARN & RECORD

Abstraction & Reusable Procedures

I can: Use a meaningful procedure name to hide details without changing expected behavior.

Website: lesson 1.4
CodeHS: 1.8.1 Abstraction; 1.8.3 Abstracting Away build_tower; 1.8.4 Abstracting Your Day; 1.7.4 The Two Towers + Comments
AP Classroom: 1.3; preview 3.12 / 3.13 (assigned AP Daily / topic questions)

Define the vocabulary in your own words

Term	Your definition + example
abstraction	_____
procedure call	_____
implementation	_____
interface	_____

Guided video and website notes

1 **How is abstraction different from decomposition?**

2 **What should a procedure name communicate?**

3 **What must remain true when you replace detailed steps with a procedure?**

One rule and an illustration to remember

Add a rule, small diagram, or example that you can use on a later assignment.

LESSON 1.4 • VISUALIZE & PLAN

Abstraction & Reusable Procedures

Read the model. Predict or draw before looking at the worked solution on the website.

Modeled problem

Instead of MOVE, MOVE, LEFT, MOVE, RIGHT, use GO_AROUND_CORNER().

A **Describe what happens step by step. Label every important state or decision.**

Diagram your reasoning

Use boxes, arrows, and labels. Include the initial state, intermediate decisions, and the result.

B **Write the corresponding algorithm in readable pseudocode. Mark input, process and output when relevant.**

STRIDES check

Which illustration or rule from today would be useful when you encounter a similar task later?

LESSON 1.4 • APPLY & REFLECT

Abstraction & Reusable Procedures

Complete the assigned CodeHS activity and the matching website investigation. Predict before running.

Transfer challenge

Create two levels of detail for “set up a study session,” showing a top-level plan and one expanded procedure.

Plan → test → revise

A **Write your initial algorithm or solution before using a computer.**

Test / starting state	Expected outcome	Observed outcome
_____	_____	_____
_____	_____	_____
_____	_____	_____

B **What did the first test reveal? Describe a change or explain why no revision was needed.**

C **Why is a procedure with a misleading name a poor abstraction?**

☐ Website notes & checks ☐ CodeHS activities ☐ AP Daily/topic work ☐ Workbook

LESSON 1.5 • LEARN & RECORD

Errors, Debugging & Corrections

I can: Use evidence to identify syntax, runtime, and logic errors and justify fixes.

Website: lesson 1.5
CodeHS: 1.2.6 Dancing Karel; 1.15.1 Debugging Basics; 1.15.3 Debug: The Two Towers; 1.15.5 Debugging From Error Messages
AP Classroom: 1.4 (assigned AP Daily / topic questions)

Define the vocabulary in your own words

Term	Your definition + example
syntax error	_____
runtime error	_____
logic error	_____
debugging	_____

Guided video and website notes

1 **Can a running program still contain a bug?**

2 **Why record expected and actual results?**

3 **Which input best tests a “70 or higher” cutoff?**

One rule and an illustration to remember

Add a rule, small diagram, or example that you can use on a later assignment.

LESSON 1.5 • VISUALIZE & PLAN

Errors, Debugging & Corrections

Read the model. Predict or draw before looking at the worked solution on the website.

Modeled problem

IF score > 70 DISPLAY "PASS" ELSE DISPLAY "RETRY". Requirement is “70 or higher passes.”

A **Describe what happens step by step. Label every important state or decision.**

Diagram your reasoning

Use boxes, arrows, and labels. Include the initial state, intermediate decisions, and the result.

B **Write the corresponding algorithm in readable pseudocode. Mark input, process and output when relevant.**

STRIDES check

Which illustration or rule from today would be useful when you encounter a similar task later?

LESSON 1.5 • APPLY & REFLECT

Errors, Debugging & Corrections

Complete the assigned CodeHS activity and the matching website investigation. Predict before running.

Transfer challenge

A robot tries to move through a wall and crashes. Describe the error, a diagnostic test, and a safe revision.

Plan → test → revise

A **Write your initial algorithm or solution before using a computer.**

Test / starting state	Expected outcome	Observed outcome
_____	_____	_____
_____	_____	_____
_____	_____	_____

B **What did the first test reveal? Describe a change or explain why no revision was needed.**

C **Why should a debugger change one cause at a time?**

☐ Website notes & checks ☐ CodeHS activities ☐ AP Daily/topic work ☐ Workbook

LESSON 1.6 • LEARN & RECORD

Testing, Tracing & Counterexamples

I can: Trace changing state and design normal, boundary, and unusual test cases.

Website: lesson 1.6
CodeHS: 1.15.2 Debugging Basics; 1.16.1 Karel Algorithms; 1.16.2 Quiz: Which Control Structure?; 1.19.1 Introduction to Programming Quiz (when ready)
AP Classroom: 1.4; preview 3.5 / 3.9 (assigned AP Daily / topic questions)

Define the vocabulary in your own words

Term	Your definition + example
trace table	
boundary case	
test case	
counterexample	

Guided video and website notes

1 **Does one passing test prove correctness for all inputs?**

2 **What changes in a trace table after x ← x + 2?**

3 **Why test values near a cutoff?**

One rule and an illustration to remember

Add a rule, small diagram, or example that you can use on a later assignment.

LESSON 1.6 • VISUALIZE & PLAN

Testing, Tracing & Counterexamples

Read the model. Predict or draw before looking at the worked solution on the website.

Modeled problem

score ← 2; score ← score + 3; score ← score * 2; IF score > 9: score ← score - 4; DISPLAY score.

A **Describe what happens step by step. Label every important state or decision.**

Diagram your reasoning

Use boxes, arrows, and labels. Include the initial state, intermediate decisions, and the result.

B **Write the corresponding algorithm in readable pseudocode. Mark input, process and output when relevant.**

STRIDES check

Which illustration or rule from today would be useful when you encounter a similar task later?

LESSON 1.6 • APPLY & REFLECT

Testing, Tracing & Counterexamples

Complete the assigned CodeHS activity and the matching website investigation. Predict before running.

Transfer challenge

Design a five-case test plan for a program admitting ages 13 through 17 inclusive.

Plan → test → revise

A **Write your initial algorithm or solution before using a computer.**

Test / starting state	Expected outcome	Observed outcome
_____	_____	_____
_____	_____	_____
_____	_____	_____

B **What did the first test reveal? Describe a change or explain why no revision was needed.**

C **Give a counterexample to “the program works for every allowed input.”**

☐ Website notes & checks ☐ CodeHS activities ☐ AP Daily/topic work ☐ Workbook

LESSON 1.7 • LEARN & RECORD

Efficiency, Heuristics & Limits (Preview)

I can: Compare growth of work and explain heuristic vs optimal solutions.

Website: lesson 1.7
CodeHS: 1.16.1 Karel Algorithms (support only); use site Efficiency Lab D for this topic
AP Classroom: 3.17 / 3.18 preview; revisit Unit 5 (assigned AP Daily / topic questions)

Define the vocabulary in your own words

Term	Your definition + example
algorithmic efficiency	_____
heuristic	_____
reasonable time	_____
undecidable problem	_____

Guided video and website notes

1 **What happens to n^2 work when n doubles?**

2 **Does a heuristic always find the shortest route?**

3 **Is exponential time the same as undecidability?**

One rule and an illustration to remember

Add a rule, small diagram, or example that you can use on a later assignment.

LESSON 1.7 • VISUALIZE & PLAN

Efficiency, Heuristics & Limits (Preview)

Read the model. Predict or draw before looking at the worked solution on the website.

Modeled problem

For n=10, linear=10, quadratic=100, exponential=1024 operations.

A **Describe what happens step by step. Label every important state or decision.**

Diagram your reasoning

Use boxes, arrows, and labels. Include the initial state, intermediate decisions, and the result.

B **Write the corresponding algorithm in readable pseudocode. Mark input, process and output when relevant.**

STRIDES check

Which illustration or rule from today would be useful when you encounter a similar task later?

LESSON 1.7 • APPLY & REFLECT

Efficiency, Heuristics & Limits (Preview)

Complete the assigned CodeHS activity and the matching website investigation. Predict before running.

Transfer challenge

Compare nearest-next-stop routing with checking every ordering. Explain an advantage and a limitation of nearest-next-stop.

Plan → test → revise

A **Write your initial algorithm or solution before using a computer.**

Test / starting state	Expected outcome	Observed outcome
_____	_____	_____
_____	_____	_____
_____	_____	_____

B **What did the first test reveal? Describe a change or explain why no revision was needed.**

C **Why do we revisit efficiency in Unit 5?**

☐ Website notes & checks ☐ CodeHS activities ☐ AP Daily/topic work ☐ Workbook

UNIT INVESTIGATION

Design brief — Campus Delivery Robot

> **Scenario:** A small robot transports a folder from the office to a classroom through corridors. It must avoid walls and report completion. You will design, test, and justify the solution before coding.

Identify the purpose and users

1 **Who is the user and what problem does the robot solve?**

2 **List the starting state, allowed commands, possible barriers and desired output.**

Input → process → output

Show the information the robot receives, the steps it performs, and the successful result.

3 **Name at least one failure case the first design must detect.**

UNIT INVESTIGATION

Design your robot algorithm

Use top-down decomposition. Your plan must include at least one named procedure that hides a sequence of individual commands.

Top-down structure chart

Place DELIVER_FOLDER at the top. Add 3–5 major subtasks and expand one subtask.

Readable pseudocode

Write a beginning, ordered procedures, at least one check, and an ending.

1 **How does your abstraction simplify the main algorithm?**

UNIT INVESTIGATION

Test record and revision

Run three different starting conditions in the website Precision Robot lab or an equivalent CodeHS Karel world. Record what you predicted and what happened.

Start state / obstacle	Predicted result	Actual result	Pass?

Bug report

1 **What was the first failure or limitation you found? Identify the evidence.**

2 **What did you change? Why should this change correct the cause?**

3 **What additional test would challenge your revised algorithm?**

UNIT INVESTIGATION

Reason about efficiency and design choices

> Compare two routes or two algorithms. They may both work without requiring the same number of commands.

Solution	Moves	Turns	Total commands	Works?
Option A				
Option B				

1 **Which design would you choose and why? Include a tradeoff beyond the command count.**

2 **How might a nearest-next-stop heuristic behave if there were several destinations?**

3 **Write a short reflection connecting purpose, abstraction, testing and improvement.**

UNIT REVIEW

Vocabulary, definitions and misconceptions

Complete without referring to the notes first

Concept	Your explanation or example
algorithm	_____
precondition	_____
purpose vs behavior	_____
decomposition	_____
abstraction	_____
logic error	_____
boundary case	_____
heuristic	_____

1 **Explain why a program can be correct for its declared preconditions but fail from a different starting state.**

2 **Explain why a heuristic is not the same as an exact optimal algorithm.**

UNIT REVIEW

AP-style conceptual reasoning

These are independently written AP-style practice questions, not College Board released items. Choose and explain an answer.

1 **A system correctly displays local weather but users need severe-weather alerts. Is the problem with implementation or purpose? Explain.**

2 **A robot runs MOVE, LEFT, MOVE. Which is a precondition: “start facing east” or “end facing north”? Why?**

3 **A function SORT_NAMES hides a long algorithm. What is gained and what still must be tested?**

4 **A pass threshold is 80 inclusive. Why are 79, 80 and 81 more informative than 50, 60 and 90?**

AP reminder

Unit 1 focuses chiefly on Creative Development topics 1.1–1.4. The algorithmic efficiency material is a preview of Topics 3.17–3.18, revisited in Unit 5.

UNIT REVIEW

Independent tracing and debugging

> Do not use a computer until you have completed your first prediction.

```
count ← 1
count ← count + 4
IF (count ≥ 5)
{
    count ← count * 2
}
DISPLAY(count)
```

Step	Statement executed	count
1	count ← 1	
2	count ← count + 4	
3	condition checked	
4	final assignment	

1 **What is displayed? Show each value of count.**

2 **Change exactly one symbol so that the body does not execute when count is 5.**

3 **A classmate says every program that terminates must be correct. Give a counterexample.**

UNIT REVIEW

My Unit 1 reference sheet + submission

Build a study tool that you can use again during AP review. Keep the examples concise and original.

The most important rules I learned

Record five rules or conventions you want to remember.

My best algorithm diagram

Draw one small input/process/output diagram OR decision flowchart.

Example pseudocode & a test case

Write the algorithm and predict a test result.

☐ All seven lessons ☐ Four labs ☐ CodeHS tasks ☐ Topic review ☐ Workbook ready to turn in