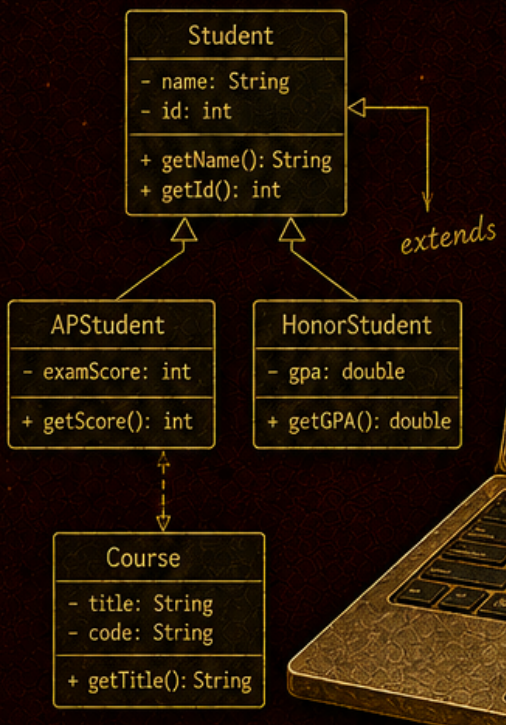
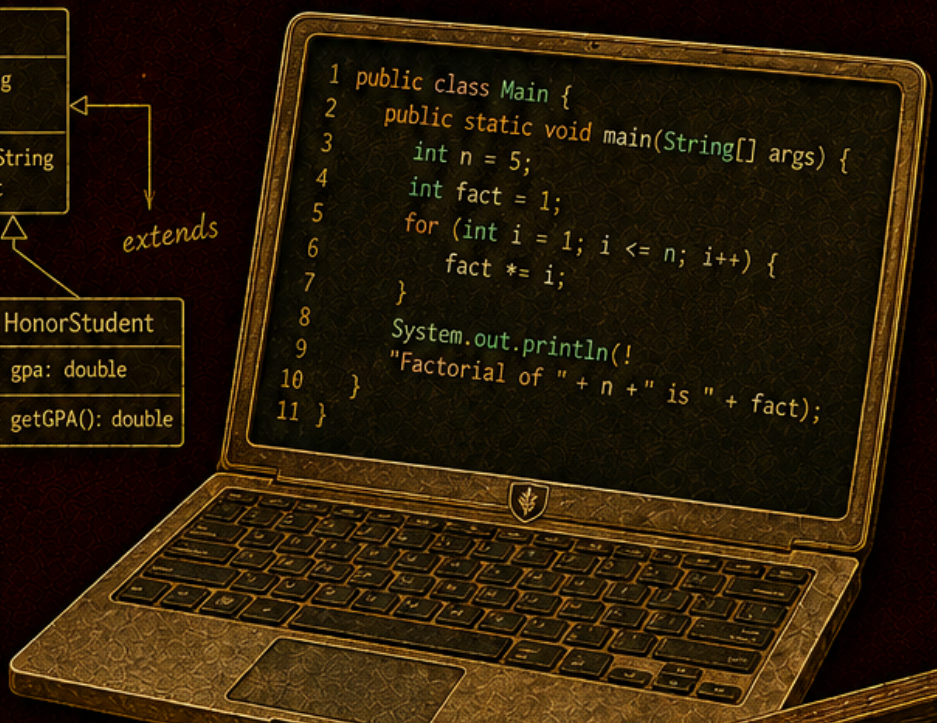
AP COMPUTER SCIENCE A

100 AI PROMPTS

for Smarter Revision *and* Exam Prep

Active recall, exam technique,
and free-response thinking — without cheating.

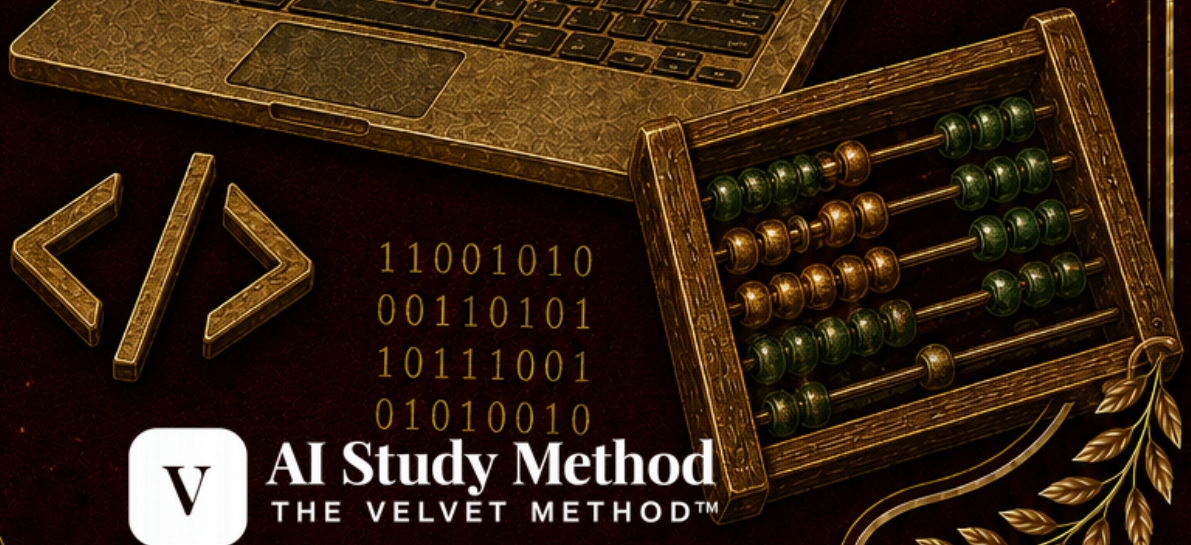




1011001
0100110
1101011
0011101
1010010

01101001
10110110
01011010
11010011
00101101

11001010
00110101
10111001
01010010



AI Study Method
THE VELVET METHOD™

THE VELVET METHOD™

A six-step AI study system

AP Computer Science A

100 AI Prompts for Smarter Revision & Exam Prep

Active recall, exam technique, and mark-scheme thinking — without cheating.

aistudymethod.co.uk

First published 2026 · aistudymethod.co.uk

What Is the Velvet Method?

The Velvet Method is a six-step framework for using AI as a study partner — not a shortcut. Each letter is a stage of effective revision, and each is grounded in what cognitive science says actually works: building understanding, retrieving it under pressure, and spacing your review over time.

V — View See the whole picture of a topic before you study the detail.

E — Evaluate Run a diagnostic to find exactly where your gaps are.

L — Learn Fill those gaps in a way that genuinely sticks.

V — Verify Prove you know it with exam-style questions and examiner-style feedback.

E — Explore Push into top-grade thinking: application, analysis, and unseen problems.

T — Transform Reflect, plan your next session, and lock it in with spaced repetition.

Used in order, the six stages turn AI from a machine that hands you answers into a tutor that strengthens your own thinking — so the work stays yours, and so does the understanding you carry into the exam.

How This Book Is Organised

This book follows the Velvet Method exactly. Its six chapters are the six stages, and the prompts in each chapter do that stage's job for AP Computer Science A. View prompts help you map the course; Evaluate prompts diagnose your gaps; Learn prompts help you understand; Verify prompts test you against mark schemes; Explore prompts stretch you into top-grade thinking; and Transform prompts help you reflect and remember.

Work through the stages roughly in order when you tackle a whole topic, or dip into a single stage when you need it — a quick diagnostic before a test, or some examiner-style marking the night before.

Every prompt is a starting point, not a script. Adapt them, raise the difficulty, swap in your own topic, and ask the AI to challenge your thinking rather than do it for you. The goal is never to get answers from AI. It is to use AI to test, challenge, and sharpen what you know.

How to Run a Velvet Session

Having a hundred prompts is not the same as knowing how to use them. Here is how the six stages fit together in a real study session — and how to turn them into a habit rather than a one-off.

For a new topic, work roughly in order: View to map it, Evaluate to find your gaps, Learn to fill the most important ones, Verify to prove they're fixed, Explore to stretch, and Transform to lock it in and plan your next review. You won't always use all six — before a test you might go straight to Evaluate and Verify.

A worked example

Say you have ninety minutes on classes and objects (Units 2 and 5). Open with a View prompt to map the sub-topics — class vs object, instance variables, constructors, accessor and mutator methods, this keyword, static vs instance, encapsulation. Run the Evaluate diagnostic and find that designing a class from a specification and using the this keyword correctly are shaky. Use two Learn prompts: an analogy for what 'instance variables encapsulate state', and a Walk-Me-Through prompt for writing a complete class with constructor and accessor/mutator methods. Then Verify with an examiner-marked FRQ 2 (class design) on implementing a class from a specification. Finally, a Transform prompt schedules a review of class structure.

Spread it out

The method works partly because it spaces your learning, so resist racing through this whole book in one sitting. A realistic rhythm is one or two topics a week through the full cycle, with short Verify and Transform reviews of earlier topics layered on top. The spacing is not a delay — it is the mechanism.

Lean on the prompts less over time

At first, use the prompts word for word. As you grow more confident, strip the scaffolding away — shorten them, raise the difficulty, and eventually run a whole stage from memory. The aim is to need this book less and less.

Trust, but Verify

AI is a powerful study partner, but it is not always right — and the danger is that you are using it precisely because you don't yet know the material well enough to notice when it's wrong. Treating its output as automatically correct is the single biggest risk in studying with AI.

What to watch for in AP Computer Science A especially: AI can invent CED unit numbers and learning objectives that don't appear in the official Course and Exam Description; it can confuse free-response rubric wording (College Board rubrics are very specific about what earns a point); it can blur the line between AP content and college-level content beyond the syllabus; and it can quote scoring percentages or score distributions that are out of date. It can use Java idioms beyond the AP Java subset (lambda expressions, streams, generics beyond ArrayList), produce code with off-by-one errors that compile but fail tests, or use the wrong rubric language. Always cross-reference with the official AP CED and recent free-response scoring guidelines.

So build the habit of checking. Keep your specification and a trusted textbook to hand, and cross-reference anything that will go into your memory. Ask the AI to show its reasoning and where a mark point comes from, so you can judge it. When a marked answer surprises you, challenge it — ask why that is the model answer. And use the Verify-stage prompt that asks the AI to include a deliberate error, to keep your own eye sharp. If the AI and your specification ever disagree, the specification wins.

This is not a reason to distrust the tool. It is the skill that makes you safe to use it — and learning to check AI well is itself one of the most valuable things this method will teach you.

Quick Start: Which Prompt, When

Not sure where to begin? Match your situation to a stage:

Starting a brand-new topic: View (Prompts 1–10), then run a diagnostic.

A test tomorrow, short on time: the diagnostics (Prompts 11–25), then a few Verify prompts on your weak spots.

You've revised but it won't stick: Learn (Prompts 26–43), especially Feynman and draw-from-memory.

You know it but keep losing exam marks: Verify (Prompts 44–71), especially examiner marking and command words.

Aiming for the top grades: Explore (Prompts 72–88).

End of a session or week: Transform (Prompts 89–100).

When in doubt, run the Whole-Topic Diagnostic (Prompt 11) — it will tell you what you need.

These prompts are your toolkit.

The Velvet Method course is where you learn to wield them.

Six steps taught in full, worked examples for every stage, and a complete prompt library across your subjects.

Learn the method at aistudymethod.co.uk/courses

Contents

Note on Exam Boards and Syllabi

This book supports AP Computer Science A students preparing for the College Board's exam, scored on the 1-5 scale. AP CS A is a Java-based course.

AP Computer Science A is organised by the College Board around 10 Units (Primitive Types; Using Objects; Boolean Expressions and if Statements; Iteration; Writing Classes; Array; ArrayList; 2D Array; Inheritance; Recursion). Five Computational Thinking Practices are assessed: Program Design and Algorithm Development, Code Logic, Code Implementation, Code Testing, Documentation. The exam is 90 minutes Multiple Choice (40 questions, 50%) plus 90 minutes Free Response (4 questions, 50%). All exam code is in Java.

Always cross-reference with your own specification and use it as the definitive guide to what you will be examined on. This book is a powerful revision companion; your specification remains the final word on content.

V

View

Before you revise a single detail, build the big picture. The View stage uses AI to map the syllabus, find the core ideas a topic depends on, and see how everything connects — so that when you study the detail, it has somewhere to attach. Start here for any new topic.

Prompt 1: Map the Whole Specification

Copy this prompt into your AI tool:

> Act as an AP Computer Science tutor. Give me a structured overview of the entire AP Computer Science course, grouped into its major areas: fundamentals of programming, data structures, algorithms and complexity, theory of computation, computer architecture, system software and operating systems, networks and the internet, databases, ethical/legal/social issues, and Boolean and number systems. For each area, list the key sub-topics in a single line.

What this helps you practise: Seeing the whole course as one connected map before revising any single part.

How to use it well: Keep this overview somewhere visible and tick off each sub-topic as you secure it, so you always know where a new technique fits in the bigger picture.

Prompt 2: Build the Big Picture of a Topic

Copy this prompt into your AI tool:

> Before I study [topic, e.g. computer networks, or theory of computation] in detail, give me a one-paragraph big-picture summary: what this topic is fundamentally about, the three or four core ideas everything else hangs on, and how it connects to the rest of AP Computer Science.

What this helps you practise: Building a mental framework that later detail can attach to.

How to use it well: Read this before your textbook, not after. The framework makes the detailed reading far easier to absorb.

Prompt 3: Create a Personalised Spec Checklist

Copy this prompt into your AI tool:

> Using the AP Computer Science Course and Exam Description (CED) for [the AP] on the topic of [topic], produce a checklist of everything I must be able to do, written as a list of 'I can…' statements (for example, 'I can trace the execution of a recursive merge sort on a small array and state its time complexity').

What this helps you practise: Turning a vague topic into a concrete, testable list of skills.

How to use it well: Tell the AI you are studying for the College Board's AP exam. Then RAG-rate each statement (red/amber/green) so your revision targets reds first.

Prompt 4: Find the Spine of a Topic

Copy this prompt into your AI tool:

> Identify the five or six absolutely central ideas in [topic, e.g. algorithms, or networks] that everything else depends on. Explain in one line why each is load-bearing, and what falls apart if I don't understand it.

What this helps you practise: Distinguishing the core principles from the peripheral detail.

How to use it well: Master the spine first. Once the central ideas are secure, the surrounding methods attach quickly and make sense.

Prompt 5: See How a Topic Connects to the Course

Copy this prompt into your AI tool:

> Show me how [topic] links to other areas of AP Computer Science. Give me a cross-unit map: for each connection, name the other topic and explain the link in one sentence (for example, data structures link to algorithms via complexity, to memory layout via pointers, and to databases via indexing).

What this helps you practise: Thinking synoptically, which is what separates top grades at AP.

How to use it well: Synoptic links are heavily rewarded. Build these maps early so that by exam season the connections feel natural rather than forced.

Prompt 6: Prioritise by Exam Weighting

Copy this prompt into your AI tool:

> Based on past AP Computer Science papers for [the AP], tell me which sub-topics of [module, e.g. paper 1 or paper 2] appear most frequently and carry the most marks. Help me rank what to prioritise if my revision time is limited.

What this helps you practise: Spending limited time where it earns the most marks.

How to use it well: Treat this as guidance, not gospel — always cross-check against your Course and Exam Description (CED). But it's a sensible way to triage when time is short.

Prompt 7: Get a Five-Minute Primer Before a Lesson

Copy this prompt into your AI tool:

> I'm about to study [topic, e.g. finite state machines] for the first time. Give me a five-minute primer: the key vocabulary I'll meet (state, transition, alphabet, accept state), the main idea in plain English, and the three questions I should be able to answer by the end of the lesson.

What this helps you practise: Priming your brain so new teaching lands on prepared ground.

How to use it well: Previewing before a lesson dramatically improves how much you take from it. The questions also tell you what to listen for.

Prompt 8: Turn Spec Points into Exam Questions

Copy this prompt into your AI tool:

> Take these Course and Exam Description (CED) statements for [topic] and, for each one, write the kinds of exam questions an AP grader could realistically ask from it, including the likely command words (state, explain, write pseudocode, trace, compare) and mark allocations.

What this helps you practise: Seeing your syllabus through the AP grader's eyes.

How to use it well: This converts passive spec points into active targets. Keep the questions to test yourself with later in the Verify stage.

Prompt 9: Sketch a Stack-of-Layers Map

Copy this prompt into your AI tool:

> Describe a layered stack I should be able to draw and label by heart for AP Computer Science — for example, the TCP/IP stack, or the layered abstraction from application code down through OS, instruction set, and hardware. For each layer, give one defining responsibility and one example. Present them as nodes for me to draw out by hand.

What this helps you practise: Organising layered concepts into one connected map.

How to use it well: Draw it yourself rather than copying — the act of building the stack is where the learning happens. Add example protocols or instructions as a second pass.

Prompt 10: Set a Clear Goal for a Revision Session

Copy this prompt into your AI tool:

> Help me set a specific, achievable objective for a 45-minute revision session on [topic, e.g. Boolean algebra simplification]. Define exactly what I should be able to do by the end, and break it into three concrete checkpoints.

What this helps you practise: Giving each session a clear purpose instead of vague 'practice'.

How to use it well: A session with a defined endpoint is far more effective than open-ended revision. Check off the three checkpoints as you go.

E

Evaluate

You can't fix gaps you haven't found. The Evaluate stage uses AI as a diagnostic: it tests you, catches the things you only think you know, and tells you precisely where to spend your time. Honest answers here save you hours later.

Prompt 11: Whole-Topic Diagnostic with Confidence Rating

Copy this prompt into your AI tool:

> You are an AP Computer Science AP grader running a diagnostic. Write 12 short questions spanning [topic, e.g. algorithms], with a spread of difficulty and varied command words (state, trace, explain, write pseudocode). Present them and wait. I will answer each and tag my confidence as (Sure), (Unsure) or (Guess). Then mark against concise model answers, flag anything I marked (Sure) but got wrong as my most dangerous gap, flag anything I got right but marked (Guess) as not yet secure, and rank my weakest sub-topics with what to study first. Finally, give me a one-sentence summary of my top three dangerous gaps that I can paste straight into my revision tracker. Before you write the questions, ask me to predict which two or three sub-topics I expect to find hardest. At the end, compare my prediction with where I actually struggled — the gap between the two is what I most need to calibrate.

What this helps you practise: Pinpointing exactly where your understanding is weakest before you revise.

How to use it well: The confidence tags are the magic: they catch the gaps a normal quiz hides — the things you got right by luck and the things you were wrong but certain about.

Prompt 12: Rapid Recall Audit

Copy this prompt into your AI tool:

> Give me 15 quick recall questions covering the breadth of [module, e.g. computer architecture], one at a time, no notes allowed. Mix definitions, conceptual checks and short pseudocode reading. At the end, sort every topic I touched into three buckets: secure, shaky, and missing.

What this helps you practise: A fast, honest snapshot of what's actually in your memory.

How to use it well: Do this closed-book and resist the urge to look anything up — the value is in the honest result, not a good score.

Prompt 13: Find My Misconceptions

Copy this prompt into your AI tool:

> Ask me to explain [concept, e.g. the difference between a stack and a heap, or how IP and MAC addresses work together]. As I answer, probe for the specific misconceptions AP Computer

> Science students commonly hold (for example, thinking the heap is part of the stack, or that IP addresses are permanent), and tell me clearly which ones I'm showing.

What this helps you practise: Surfacing the subtle wrong ideas that quietly cost marks.

How to use it well: Misconceptions are dangerous because they feel like knowledge. Naming yours is the first step to fixing them.

Prompt 14: Pre-Test Triage

Copy this prompt into your AI tool:

> I have a test on [topic] tomorrow. Run a quick diagnostic to find out where my understanding is weakest, then tell me precisely which two or three areas to spend tonight on, and roughly how long on each.

What this helps you practise: Turning night-before panic into a focused, prioritised plan.

How to use it well: Be honest in your answers — a diagnosis based on bluffing sends you to revise the wrong things.

Prompt 15: Command-Word Diagnostic

Copy this prompt into your AI tool:

> Test whether I can handle different command words on the same content. Ask me a 'state', then a 'describe', then an 'explain', then a 'compare' question on [topic, e.g. paging vs segmentation], and tell me which command word I struggle with most.

What this helps you practise: Diagnosing exam-technique gaps, not just knowledge gaps.

How to use it well: 'Describe' wants what happens; 'explain' wants why. Knowing the difference saves marks examiners predictably remove.

Prompt 16: Map the Gaps Across a Module

Copy this prompt into your AI tool:

> Sweep across the whole of [module, e.g. data structures and algorithms] with a broad set of questions including tracing, complexity analysis, conceptual explanation and short pseudocode. Output a ranked gap list — my weakest area at the top — with a one-line reason for each ranking.

What this helps you practise: Getting a module-wide picture rather than topic tunnel vision.

How to use it well: Use the ranked list to plan a week of revision, working top-down from your biggest weakness.

Prompt 17: Test Depth, Not Just Surface

Copy this prompt into your AI tool:

> Ask me about [concept, e.g. why a hash table has expected O(1) lookup but worst-case O(n)]. Whatever I answer, keep asking 'why' and 'how do you know' until either my understanding runs out or you're satisfied it's genuinely deep. Then tell me where it ran out.

What this helps you practise: Distinguishing real understanding from memorised statements.

How to use it well: The point where you can no longer answer 'why' is exactly where your real revision should begin.

Prompt 18: Diagnose My Pseudocode Fluency

Copy this prompt into your AI tool:

> Test my pseudocode fluency for [the AP]'s conventions with a short mixed set: a loop with a sentinel value, a function with parameters and a return value, a recursive function, and an example using a stack. Tell me which I need to practise.

What this helps you practise: Catching the pseudocode weaknesses that quietly cost marks across the paper.

How to use it well: Different boards have different pseudocode conventions. Use the AP's reference language — don't write Python and hope.

Prompt 19: Diagnose My Theory-of-Computation Skills

Copy this prompt into your AI tool:

> Quiz me on the theory-of-computation content: finite state machines, regular expressions, BNF grammars, Turing machines and undecidability. Where I get something wrong, identify whether the cause is notation, definition, or application. Tell me where my understanding is thinnest.

What this helps you practise: Securing the under-revised theory-of-computation marks examiners predictably set.

How to use it well: Theory of computation is heavily examined and lightly revised. A short, focused diagnostic here pays back disproportionately.

Prompt 20: Test a Linked Concept

Copy this prompt into your AI tool:

> Give me a diagnostic that specifically checks whether I can connect [topic A, e.g. memory management] and [topic B, e.g. virtual memory and paging], not just recall each separately. Tell me whether my weakness is in the individual topics or in linking them.

What this helps you practise: Diagnosing cross-unit weakness, a common hidden gap.

How to use it well: You can know two topics well and still fail to link them. This isolates which problem you actually have.

Prompt 21: Self-Explanation Check

Copy this prompt into your AI tool:

> I'm going to explain [topic, e.g. how TCP guarantees reliable delivery] to you in my own words. Listen, then identify every gap, error, vague statement, or missing link in my explanation, in order of importance.

What this helps you practise: Forcing the gaps in your understanding to reveal themselves.

How to use it well: Explaining out loud exposes far more than re-reading ever will. The vague bits in your explanation are your real gaps.

Prompt 22: Exam-Readiness Score

Copy this prompt into your AI tool:

> Act as a tough AP Computer Science AP grader. Ask me the hardest questions you realistically could on [topic], then rate my readiness for each sub-area on a scale of 1 to 5 and justify each score.

What this helps you practise: Getting a frank, AP grader's-eye assessment of where you stand.

How to use it well: Ask it to be strict. A flattering score the week before an exam helps nobody.

Prompt 23: Confidence vs Competence Audit

Copy this prompt into your AI tool:

> For [topic, e.g. networks], first ask me how confident I feel about each sub-area (protocols, OSI/TCP-IP layers, packet switching, security). Then test me on each. Finally, show me where my confidence and my actual performance don't match.

What this helps you practise: Exposing over-confidence, which is where careless marks are lost.

How to use it well: The mismatches matter most: areas you feel sure about but perform poorly on are where exam disasters happen.

Prompt 24: Diagnose from a Mistake

Copy this prompt into your AI tool:

> Here is a question I got wrong: [paste the question and my answer]. Don't just give me the right answer — diagnose the underlying gap or misconception that caused the mistake, and tell me what else that gap probably affects.

What this helps you practise: Learning from errors at the level of the cause, not the symptom.

How to use it well: One mistake often points to a deeper gap that's costing you elsewhere too. Fix the cause, not just the instance.

Prompt 25: Weekly Diagnostic Sweep

Copy this prompt into your AI tool:

> Run my weekly diagnostic across everything I've revised this week: [list topics]. Compare my performance, tell me what has improved and what is slipping, and what needs revisiting before next week.

What this helps you practise: Tracking progress and catching topics that fade over time.

How to use it well: Knowledge decays without review. A weekly sweep catches the slipping topics before they're forgotten.

L

Learn

Now fill the gaps in a way that sticks. The Learn stage uses AI to explain, analogise, roleplay, and break concepts down — and crucially to check your understanding, not just deliver information. One rule for every prompt here: when an explanation finally clicks, close the chat and reproduce it from memory, out loud or on paper, before moving on. Understanding something as you read it is not the same as being able to recall it later — the moment of 'I get it' is the cue to test yourself, not to stop.

Prompt 26: Explain with an Analogy

Copy this prompt into your AI tool:

> Explain [difficult concept, e.g. recursion, or how a CPU pipeline works] using a clear everyday analogy. Then ask me to map each part of the analogy back onto the real computer science to check I've understood, not just enjoyed the story.

What this helps you practise: Grasping an abstract concept by anchoring it to something familiar.

How to use it well: Always map the analogy back to the computer science — an analogy you can't translate is entertainment, not understanding.

Prompt 27: Use the Feynman Technique

Copy this prompt into your AI tool:

> I'm going to explain [topic, e.g. how a stack frame is used in a function call] as simply as if I were teaching a 14-year-old. Listen, then point out exactly where I used jargon to paper over a gap, where my logic broke, and where I oversimplified to the point of being wrong. When you spot a gap in my explanation, do not correct me immediately. Give me one 60-second attempt to re-explain that part from memory first; only after my retry do you give the correction. This iteration is where the encoding consolidates.

What this helps you practise: Achieving genuine understanding by teaching it plainly.

How to use it well: If you can't explain it simply, you don't fully understand it. The jargon you reach for marks the gaps.

Prompt 28: Roleplay a Packet

Copy this prompt into your AI tool:

> Roleplay as a TCP packet travelling from a client browser to a web server. Narrate what happens at each layer — application, transport, internet, link — and quiz me on what's happening as we go, including encapsulation, addressing and the three-way handshake.

What this helps you practise: Making the layered network stack vivid and memorable.

How to use it well: Narrative ordering helps multi-step protocols stick. Try it for any layered process where the sequence matters.

Prompt 29: Build a Mnemonic

Copy this prompt into your AI tool:

> Help me create a memorable mnemonic for [ordered set, e.g. the OSI seven layers, or the Fetch–Decode–Execute cycle stages]. Then test me on whether I can reconstruct the full list from the mnemonic.

What this helps you practise: Locking ordered sequences into long-term memory.

How to use it well: Make the mnemonic slightly absurd or personal — odd associations are easier to recall than bland ones.

Prompt 30: Trace, Then Blank It Out

Copy this prompt into your AI tool:

> Trace [algorithm, e.g. quicksort on the array [4,7,2,8,1,3]] one step at a time, showing the state of every variable and recursive call. Then give the same trace back to me with key rows blanked out for me to fill in.

What this helps you practise: Moving from guided understanding to independent algorithm tracing.

How to use it well: Tracing is retrieval. Cover the model trace and reproduce it on your own — that's where the learning happens.

Prompt 31: Compare and Contrast

Copy this prompt into your AI tool:

> Build a comparison table for [X vs Y, e.g. array vs linked list, or stack vs queue]. Include the points AP examiners care about: memory layout, insertion and deletion complexity, real-world use cases. Then quiz me specifically on the differences, since that's what questions target.

What this helps you practise: Sharpening the distinctions examiners love to test.

How to use it well: Confusable pairs are exam favourites. Knowing the differences cold is worth more than knowing each in isolation.

Prompt 32: Teach the Hard Bit Three Ways

Copy this prompt into your AI tool:

> Explain [tricky concept, e.g. recursion, or virtual memory] to me in three completely different ways — a definition-led way, an analogy, and a worked example. Then ask me which made it click and test that understanding.

What this helps you practise: Finding the explanation that actually works for you.

How to use it well: Different concepts yield to different explanations. Keep the version that clicks and use it as your anchor.

Prompt 33: Ground the Abstract in the Concrete

Copy this prompt into your AI tool:

> Take this abstract idea — [e.g. abstraction, or polymorphism] — and give me two concrete, specific examples from real code or real systems, then ask me to generate a third myself.

What this helps you practise: Connecting principles to the real examples exams demand.

How to use it well: Generating your own example is the test of understanding. If you can produce a new case, you've got the principle.

Prompt 34: Draw It from Memory

Copy this prompt into your AI tool:

> Guide me to draw a labelled diagram of [structure, e.g. the Von Neumann architecture, a binary tree being inserted into, or a CPU with cache and main memory] from memory. Ask me what goes where, then tell me what I missed or mislabelled.

What this helps you practise: Building the precise, labelled recall that diagrams require.

How to use it well: Drawing from memory is retrieval, not copying. Redraw until you can produce the full labelled version unaided.

Prompt 35: Why Does It Work That Way?

Copy this prompt into your AI tool:

> Don't just tell me what happens in [process, e.g. paging, or the three-way handshake] — explain why it works that way from first principles. Give the underlying reason for each step, so I understand the logic rather than memorising a sequence.

What this helps you practise: Replacing rote memory with causal understanding.

How to use it well: Understanding the 'why' means you can reconstruct the 'what' even if you forget it, and handle unfamiliar twists.

Prompt 36: Unpack a Definition Word by Word

Copy this prompt into your AI tool:

> Take this precise AP definition — [e.g. of a context-free grammar, of an algorithm, or of denormalised data] — and unpack it word by word, explaining why each term is in there and what would be wrong if it were left out.

What this helps you practise: Mastering the exact wording examiners require.

How to use it well: AP definitions are precise for a reason. Knowing why each word matters helps you reproduce them exactly.

Prompt 37: Link Data Structure to Use Case

Copy this prompt into your AI tool:

> For [data structure, e.g. hash table, binary search tree, queue], explain in detail when it is the right choice and when it is not, in terms of insertion, deletion, search complexity, and memory. Then ask me to do the same for a different structure to check I can apply the principle.

What this helps you practise: Mastering structure–use-case fit, the core skill of the data-structures topic.

How to use it well: Examiners ask 'which data structure would you choose, and why'. Practise the trade-off, not just the operations.

Prompt 38: Interleave Two Confusable Topics

Copy this prompt into your AI tool:

> Alternate between explaining and quizzing me on [topic A, e.g. pre-order tree traversal] and [topic B, e.g. in-order traversal], which I keep mixing up. Deliberately switch between them so I have to tell them apart each time.

What this helps you practise: Building the discrimination to keep similar topics separate.

How to use it well: Interleaving feels harder than studying one topic at a time, but it's exactly that difficulty that stops you confusing them.

Prompt 39: Turn My Notes into Understanding

Copy this prompt into your AI tool:

> Here are my notes on [topic, e.g. databases and normalisation]: [paste]. Turn them into a clear, causal explanation, and point out anything I've written down as a fact without actually understanding why it's true.

What this helps you practise: Converting copied notes into genuine comprehension.

How to use it well: Notes can hide a lack of understanding. This step exposes the lines you've memorised but couldn't explain.

Prompt 40: Explore What Happens If a Step Fails

Copy this prompt into your AI tool:

> For [protocol, e.g. TCP], walk through what would go wrong if one specific step failed — for example, if an ACK were lost, or if the receive window shrank to zero. Use the failure to test whether I really understand the normal working protocol.

What this helps you practise: Deepening understanding by examining how a system breaks.

How to use it well: Understanding failure modes proves you understand the working system, and prepares you for application questions.

Prompt 41: Build Precise Vocabulary

Copy this prompt into your AI tool:

> Generate the key technical terms for [topic, e.g. networks — packet, frame, segment, datagram, MAC, IP, port, socket], each with a precise AP definition and a sentence showing correct use. Then test me by giving definitions and asking for the term, and vice versa.

What this helps you practise: Building the exact vocabulary that earns marks.

How to use it well: Imprecise vocabulary loses marks even when the idea is right. Drill the exact terms until they're automatic.

Prompt 42: Explain the Behaviour of a State Machine

Copy this prompt into your AI tool:

> Talk me through interpreting a finite state machine for [scenario, e.g. parsing a binary string that ends in 01]. Explain the role of each state and transition, then give me a blank state diagram to complete from a Course and Exam Description (CED) myself.

What this helps you practise: Understanding the behaviour-level meaning of state machines, not just the diagram.

How to use it well: FSM questions reward saying what each state and transition represents in plain English alongside the diagram.

Prompt 43: Make an Algorithm into a Story

Copy this prompt into your AI tool:

> Turn [algorithm, e.g. Dijkstra's shortest-path algorithm, or merge sort] into a short, vivid story with a clear sequence of moves, then quiz me on the order of steps and what data is updated at each stage.

What this helps you practise: Using narrative structure to remember multi-step algorithms.

How to use it well: Algorithms are easier to recall as a story than as a list of steps. Make the data-update chain the plot.

V

Verify

Familiar is not the same as known. The Verify stage uses AI as an examiner: it asks exam-style questions, marks you against realistic mark schemes, and shows you exactly where marks are won and lost. This is the heart of the method, so this is the longest chapter.

Prompt 44: Examiner Marking Against a Mark Scheme

Copy this prompt into your AI tool:

> Act as a senior AP Computer Science AP grader. Give me one Short Answer question on [topic]. After I answer, mark it against a realistic mark scheme: list each available mark point, show which I hit and which I missed, award a mark out of 6, and tell me the single most valuable improvement.

What this helps you practise: Practising extended answers and learning how marks are actually awarded.

How to use it well: The mark-by-mark breakdown is the gold here. Note which mark points you keep missing — that pattern is your improvement plan.

Prompt 45: Closed-Book Retrieval Quiz

Copy this prompt into your AI tool:

> Quiz me on [topic] with ten questions, one at a time, no notes allowed. Don't give me the answer until I've committed to mine, then tell me if I'm right and correct any error precisely. Before you confirm or reject my answer, ask me to articulate in one sentence how I arrived at it. Mark me on the reasoning as well as the conclusion — a right answer from shaky reasoning is not yet secure.

What this helps you practise: Strengthening memory through the act of retrieval itself.

How to use it well: Retrieval, not re-reading, is what builds durable memory. Commit to an answer before checking — the effort is the point.

Prompt 46: Drill the 'Explain' Reasoning Chain

Copy this prompt into your AI tool:

> Give me an 'explain' question on [topic, e.g. why TCP uses a sliding window]. Mark me specifically on whether my answer forms a complete chain of reasoning, where each point causes the next, rather than a list of disconnected facts.

What this helps you practise: Building the linked reasoning that 'explain' questions demand.

How to use it well: 'Explain' means show the causal links. A list of true facts with no connections rarely scores full marks.

Prompt 47: Find the Missing Mark Points

Copy this prompt into your AI tool:

> Write a model answer to a 5-mark question on [topic], then remove two of the mark points and present it to me. Ask me to identify exactly which two mark-worthy points are missing.

What this helps you practise: Training your eye to see what a full-mark answer requires.

How to use it well: Learning to spot what's missing from a good answer is how you stop leaving marks on the table in your own.

Prompt 48: Demand Precise Definitions

Copy this prompt into your AI tool:

> Test me on the key definitions for [topic, e.g. algorithms — algorithm, complexity, recursion, abstraction]. Accept only answers that are precise to AP standard — if I'm vague or miss a key word, tell me exactly which word would have earned the mark.

What this helps you practise: Producing the exact wording that definition marks require.

How to use it well: One missing word can cost a definition mark. This drills the precision examiners insist on.

Prompt 49: Mark Pseudocode

Copy this prompt into your AI tool:

> Ask me to write a pseudocode solution to [problem, e.g. reversing a linked list, or counting word occurrences in a string] using my board's reference language conventions. Then critique my code as an AP grader would: are the variable types declared, are the control structures used correctly, do edge cases work, and is the code readable? Give me a mark out of 6.

What this helps you practise: Producing pseudocode that earns every available mark.

How to use it well: Pseudocode marks come from correctness AND board convention. Always trace your code on a small input before submitting.

Prompt 50: Application Question, Marked

Copy this prompt into your AI tool:

> Give me an application question that places [principle, e.g. abstraction, or normalisation] in an unfamiliar context. After I answer, mark whether I correctly applied the principle to the new situation rather than just restating what I memorised.

What this helps you practise: Practising the application skill that distinguishes strong candidates.

How to use it well: Application questions reward transfer, not recall. Marks come from using the principle, not reciting it.

Prompt 51: Structure a Six-Marker

Copy this prompt into your AI tool:

> Give me a Short Answer extended-response question on [topic]. Before marking my content, assess the structure of my answer: is it logically ordered, does it address the command word, and does it make the right number of distinct points for the marks available?

What this helps you practise: Structuring long answers so every mark is reachable.

How to use it well: Examiners can only award marks they can find. Clear structure makes your points easy to credit.

Prompt 52: Check a Number-System Calculation

Copy this prompt into your AI tool:

> Give me a number-system or arithmetic problem at AP level: binary to two's complement, hexadecimal addition, floating-point representation, IEEE-style normalisation, or signed-binary subtraction. Mark my working as well as my answer, since method marks matter.

What this helps you practise: Securing the number-system marks and the method marks behind them.

How to use it well: Show full working — method marks are available even when the final answer is wrong. Never just write the answer.

Prompt 53: Trace an Algorithm Under Examination

Copy this prompt into your AI tool:

> Give me an algorithm in pseudocode (sort, search, recursion, or graph) and an input. Ask me to trace it line by line, showing the state of all variables at each step. Mark me on completeness, accuracy and whether I produced a trace table examiners can follow.

What this helps you practise: Practising the trace-table skill examiners reward heavily and students under-revise.

How to use it well: Always lay traces out as a labelled table with one column per variable. Step counters earn marks; informal notes do not.

Prompt 54: Synoptic Retrieval Challenge

Copy this prompt into your AI tool:

> Ask me a question that requires me to pull together knowledge from at least three different areas of AP Computer Science to answer fully — for example, a question on database performance that needs hashing, indexing and complexity reasoning. Mark me on the breadth and the quality of the links I make.

What this helps you practise: Retrieving and connecting across topics under question pressure.

How to use it well: Synoptic questions reward range and linkage. Practise reaching deliberately into other topics for relevant points.

Prompt 55: Same Topic, Different Command Words

Copy this prompt into your AI tool:

> Ask me three questions on [topic] using three different command words — for example 'describe', 'explain' and 'compare'. Mark each against what that specific command word requires, and tell me which I handle worst.

What this helps you practise: Matching your answer to what each command word actually demands.

How to use it well: The content may overlap, but the command word changes what scores. This builds that discrimination.

Prompt 56: Mark Notational and Conventional Accuracy

Copy this prompt into your AI tool:

> Test me on writing answers in [topic, e.g. Boolean algebra simplification, or BNF grammars]. Mark me strictly on notation, ordering, and standard conventions — the technical accuracy marks examiners regularly remove without students noticing.

What this helps you practise: Hardening the notational accuracy that quietly costs marks across the paper.

How to use it well: Notation is examined separately from content. Train yourself to use the standard symbols and ordering by reflex.

Prompt 57: Spring the Common Error Trap

Copy this prompt into your AI tool:

> Ask me questions on [topic] that are deliberately designed around the errors AP Computer Science students most commonly make (for example, confusing pass-by-value and pass-by-reference, or saying 'compiler' when 'interpreter' is correct). Catch me if I fall for one and explain the precise correct version.

What this helps you practise: Inoculating yourself against the classic, predictable mistakes.

How to use it well: Examiners know the common errors and test for them. Knowing yours in advance is half the battle.

Prompt 58: Rapid-Fire Definitions

Copy this prompt into your AI tool:

> Fire 15 key terms from [module, e.g. operating systems or networks] at me, one at a time, and ask me to give the precise definition for each as fast as I can. Then list the ones I got wrong or vague for me to drill again.

What this helps you practise: Making core definitions fast and automatic.

How to use it well: Speed matters — in the exam you want definitions to come instantly so your thinking time goes on the hard parts.

Prompt 59: Predict the Mark Scheme

Copy this prompt into your AI tool:

> Show me a past-paper-style question on [topic]. Before answering, I'll predict what the mark scheme rewards (which mark points and in what wording). Then reveal a realistic mark scheme and tell me how accurate my prediction was.

What this helps you practise: Learning to think like the AP grader who wrote the mark scheme.

How to use it well: If you can predict the mark scheme, you can target your answer at it. This builds that AP grader's instinct.

Prompt 60: Verify a Logic Circuit

Copy this prompt into your AI tool:

> Ask me to draw and label the logic circuit for [Boolean expression] or to derive the expression from a circuit. Then critique my version against what an AP grader expects: correct gate symbols, correct labelling of inputs and intermediate signals, no redundant gates after simplification.

What this helps you practise: Producing exam-accurate logic circuits under recall conditions.

How to use it well: Simplify Boolean expressions before drawing. Examiners check that your simplification is at least as far as the formula sheet would allow.

Prompt 61: Drill Big-O Reasoning

Copy this prompt into your AI tool:

> Give me a problem on [topic, e.g. analysing the worst-case complexity of a nested-loop algorithm, or comparing the complexity of merge sort and bubble sort]. Mark me on the working, the dominant term, and whether I correctly stated the assumptions about the input.

What this helps you practise: Securing the complexity marks that examiners reliably award and students reliably miss.

How to use it well: Always state which input parameter n refers to and which case (best, average, worst) you're analysing.

Prompt 62: Practise a 'Compare' Question

Copy this prompt into your AI tool:

> Give me a 'compare' question on [X and Y, e.g. paging vs segmentation, or compiled vs interpreted languages]. Mark me on whether I made genuine comparative statements (linking the two) rather than describing each separately, which scores poorly.

What this helps you practise: Writing true comparisons rather than parallel descriptions.

How to use it well: A real comparison uses linking words — 'whereas', 'in contrast', 'both'. Two separate descriptions side by side rarely earn the comparison marks.

Prompt 63: Practise a 'Discuss' Question

Copy this prompt into your AI tool:

> Give me a 'discuss' question on [topic, e.g. the ethical implications of a particular technology, or the use of cloud storage vs local storage]. Mark me on whether I presented evidence on more than one side and reached a justified conclusion, rather than just listing points.

What this helps you practise: Building the balanced judgement 'discuss' requires.

How to use it well: 'Discuss' wants a weighed conclusion, not a list. Make sure you actually come down on a side, with a reason.

Prompt 64: Practise a 'Justify' Question

Copy this prompt into your AI tool:

> Give me a 'justify' question that asks me to recommend a [data structure / algorithm / hardware choice] for an unfamiliar scenario. Mark me on whether my recommendation is computationally reasonable and properly justified.

What this helps you practise: Building the applied reasoning that 'justify' questions test.

How to use it well: 'Justify' rewards reasoned recommendation in a new context — say which constraint matters and why your choice satisfies it.

Prompt 65: Climb the Define–Describe–Explain Ladder

Copy this prompt into your AI tool:

> On [single concept, e.g. a hash function], ask me first to define a key term, then to describe what happens in a worked example, then to explain why. Mark each rung separately so I can see where my answer stops being strong.

What this helps you practise: Seeing exactly where your understanding shifts from solid to shaky.

How to use it well: The rung where your answer weakens shows the boundary of your real understanding — revise from there.

Prompt 66: Mark My Real Answer

Copy this prompt into your AI tool:

> Here is my answer to [question]: [paste]. Mark it strictly to score 5 standard against a realistic mark scheme, tell me the exact mark, and rewrite one sentence of mine to show what top-mark phrasing looks like.

What this helps you practise: Getting precise, personalised feedback on your actual writing.

How to use it well: The rewritten sentence is a model to imitate. Compare it with yours to see exactly what 'top mark' phrasing adds.

Prompt 67: Fill a Retrieval Grid

Copy this prompt into your AI tool:

> Create a retrieval grid for [topic, e.g. data structures]: a set of prompts or blank cells covering the key operations, complexities, conceptual definitions and use cases, which I'll fill in entirely from memory. Then check my grid and highlight the gaps.

What this helps you practise: Wide-coverage recall practice in one efficient exercise.

How to use it well: Retrieval grids cover a lot of ground fast. Redo the gaps a day later to confirm they've stuck.

Prompt 68: Simulate a Timed Question

Copy this prompt into your AI tool:

> Give me a single AP question on [topic] with a stated mark allocation, and tell me the realistic time I'd get for it in the exam. I'll answer under that time pressure, then you mark it.

What this helps you practise: Practising working accurately at exam pace.

How to use it well: Knowing your content isn't enough if you're too slow. Practise to the clock so timing never surprises you.

Prompt 69: Sit a Mini Past-Paper Set

Copy this prompt into your AI tool:

> Give me a short set of five exam-style questions covering the breadth of [module], mixing recall, code-tracing, extended explanation and pseudocode writing, as a real paper would. Mark the whole set and summarise my strongest and weakest question types.

What this helps you practise: Rehearsing the mixed demands of a real exam paper.

How to use it well: Mixed sets reveal which question types you fear. Drill the weakest type specifically afterwards.

Prompt 70: Catch the AI's Deliberate Error

Copy this prompt into your AI tool:

> Answer this question yourself: [question, e.g. write a recursive function, or trace a sorting algorithm]. Include exactly one subtle but real error in your code or trace — for example, an off-by-one in a loop bound, a missing base case, or a wrong layer mapping in a protocol stack — and challenge me to find and correct it. Tell me if I got it, then explain the error and why it would have been easy to miss — so I learn the spotting pattern, not just the specific instance. After I respond, ask me how confident I felt before checking. The gap between my confidence and my accuracy is the most important thing for me to learn — I want to surface where I'm certain but wrong, not just where I'm wrong.

What this helps you practise: Building the critical eye to spot when AI (or you) is wrong.

How to use it well: This is essential AI-literacy: the AI can be confidently wrong, especially in edge-case code. Always trace by hand on a small input.

Prompt 71: Mixed Whole-Paper Retrieval

Copy this prompt into your AI tool:

> Quiz me with 12 questions drawn from across everything in AP Computer Science I've named: [list topics]. Mix theory, code and architecture randomly, as the papers do, and at the end tell me which topics I'm weakest on overall.

What this helps you practise: Retrieving under the topic-switching conditions of a real exam.

How to use it well: Random topic order is harder than block revision — and exactly what the exam does. Train for it.

E

Explore

Top grades come from thinking, not just knowing. The Explore stage uses AI to push you into application, analysis, unseen data, and synoptic links across the course — the level that separates good answers from excellent ones.

Prompt 72: Tackle an Unseen Algorithm

Copy this prompt into your AI tool:

> Show me an unfamiliar algorithm in pseudocode at AP level — for example, the sieve of Eratosthenes, a heap operation, or a graph traversal I haven't met. Ask me to trace it on a small input, state its complexity, and identify the design pattern (greedy, divide-and-conquer, dynamic programming), then evaluate my reasoning.

What this helps you practise: Applying algorithmic reasoning to genuinely new code, as top questions demand.

How to use it well: Always trace before you analyse. Complexity claims unsupported by tracing rarely earn the marks.

Prompt 73: Build a Synoptic Chain

Copy this prompt into your AI tool:

> Challenge me to connect at least three different AP Computer Science topics into a single chain of reasoning about [theme, e.g. how a search query is processed end-to-end through a network, OS, database and CPU]. Push me to make each link explicit and technically sound.

What this helps you practise: Constructing the extended, cross-topic reasoning that earns top grades.

How to use it well: Top candidates connect topics. Practise building chains: input → network → OS → application → storage.

Prompt 74: Apply a Principle to a New System

Copy this prompt into your AI tool:

> Take [principle, e.g. caching, or abstraction] and give me a scenario I won't have studied — an unusual system or device — where I have to apply it. Then judge whether I applied it correctly.

What this helps you practise: Transferring a principle beyond the example you learned it with.

How to use it well: Real understanding is shown by transfer. If you can only use a principle in its original example, you don't fully own it yet.

Prompt 75: Critique a System Design

Copy this prompt into your AI tool:

> Describe a flawed system design for [scenario, e.g. a poorly normalised database, or a network with no separation of concerns between layers]. Ask me to identify the weaknesses — coupling, duplication, performance bottlenecks, security holes — and suggest specific improvements. Then evaluate my critique.

What this helps you practise: Building the design-evaluation skill examiners reward heavily.

How to use it well: Design evaluation marks are abundant and under-revised. Train your eye to spot what's wrong with a structure.

Prompt 76: Predict and Justify

Copy this prompt into your AI tool:

> Set me a scenario about [topic, e.g. changing the page size in a virtual-memory system] and ask me to predict what would happen, then justify my prediction with computer-science reasoning. Tell me whether my reasoning, not just my prediction, was sound.

What this helps you practise: Reasoning forward from principles to predicted outcomes.

How to use it well: In application questions the justification scores, not the guess. Always explain the underlying mechanism behind your prediction.

Prompt 77: Investigate an Open Question

Copy this prompt into your AI tool:

> Pose me a genuinely open computer-science question related to [topic, e.g. 'is moving to immutable data structures worth the memory cost in this scenario?'] — one where you explore possibilities with me rather than giving a single answer — and challenge my reasoning at each step.

What this helps you practise: Practising the exploratory thinking behind the hardest questions.

How to use it well: Sit with open questions rather than rushing to one answer. The ability to reason through uncertainty is a top-band skill.

Prompt 78: Connect to Real Systems

Copy this prompt into your AI tool:

> Link [topic] to a real-world system or piece of widely used software — for instance Git, HTTPS, a relational database, or a modern CPU — and ask me application questions that use that context.

What this helps you practise: Seeing how textbook computer science shows up in unfamiliar exam contexts.

How to use it well: Exam contexts are often drawn from real systems. Familiar real examples make unseen contexts less intimidating.

Prompt 79: Investigate Complexity Trade-offs

Copy this prompt into your AI tool:

> Give me a problem that I could solve with [data structure / algorithm A] or with [data structure / algorithm B]. Ask me to compare the time and space complexity, the implementation effort, and the cache behaviour, then judge which I would choose and justify it.

What this helps you practise: Building the engineering-trade-off thinking that the highest-mark questions reward.

How to use it well: Top marks come from naming the trade-off explicitly. Never recommend without saying what you traded for it.

Prompt 80: Interpret an Unseen Diagram

Copy this prompt into your AI tool:

> Show me (in words) an unfamiliar diagram related to [topic] — for example a non-trivial state machine, an unfamiliar protocol exchange, or a memory layout. Ask me to interpret it from first principles, then tell me what I read well and what I missed.

What this helps you practise: Reading unfamiliar diagrams using underlying computer science, not memory.

How to use it well: Unseen diagrams test whether you understand the concepts or just memorised familiar shapes. Reason from the labels up.

Prompt 81: Reason Across Layers

Copy this prompt into your AI tool:

> Ask me a question that requires me to reason from a high-level language statement down to the CPU instructions and memory effects (or from an application-layer event down through the protocol stack to the wire) for [topic]. Judge whether I kept the chain of cause and effect intact across the layers.

What this helps you practise: Building the multi-layer reasoning that cross-unit questions demand.

How to use it well: Keeping a chain coherent across abstraction layers is a high-level skill. Practise it deliberately.

Prompt 82: Design a Database Schema

Copy this prompt into your AI tool:

> Give me a real-world scenario and ask me to design a normalised relational schema for it. Require me to state the entities, relationships, primary and foreign keys, and which normal form I've reached, then critique my design for redundancy and integrity.

What this helps you practise: Mastering database design, a guaranteed source of marks.

How to use it well: State which normal form you reached and the dependency you eliminated to reach it. Examiners reward explicit reasoning.

Prompt 83: Challenge My Reasoning

Copy this prompt into your AI tool:

> I'll make a computer-science claim and my reasoning for it about [topic, e.g. why hash tables outperform binary search trees for lookup]. Your job is to challenge it — find the flaw, the missing condition, or the better counter-argument — and push me to defend or revise my position.

What this helps you practise: Strengthening arguments by stress-testing them.

How to use it well: Having your reasoning challenged is uncomfortable and valuable. A claim that survives scrutiny is one you can defend in the exam.

Prompt 84: Choose the Right Paradigm

Copy this prompt into your AI tool:

> Give me a problem and a set of programming paradigms (procedural, object-oriented, functional, declarative). Ask me which is most appropriate and to justify the choice, then tell me if I'm right and why.

What this helps you practise: Applying paradigm-level judgement to design decisions.

How to use it well: Choosing the right paradigm is a recurring AP skill. Learn which features each paradigm makes natural.

Prompt 85: Debate an Ethical or Evaluative Issue

Copy this prompt into your AI tool:

> Set me an evaluative question on a contested area such as algorithmic bias, mass surveillance, or the environmental footprint of large-scale computing. Mark me on whether I presented a balanced argument with evidence on both sides and a justified conclusion.

What this helps you practise: Building the balanced, evidence-based judgement evaluative questions need.

How to use it well: Evaluative questions want both sides and a reasoned verdict. Avoid one-sided lists — weigh the arguments and conclude.

Prompt 86: Solve a Novel Problem

Copy this prompt into your AI tool:

> Give me a problem on [topic] that I can only solve by combining two or more principles I've learned separately — for example, a question requiring recursion AND a stack to reason about memory. Don't tell me which principles — let me work it out, then review my approach.

What this helps you practise: Combining principles to solve unfamiliar problems, the mark of mastery.

How to use it well: The hardest questions hide which knowledge to use. Practising 'which principle applies here?' is itself a skill.

Prompt 87: Apply Synoptic Thinking to a Real Application

Copy this prompt into your AI tool:

> Pick a real-world application — for example a web search engine, an online banking app, or a multiplayer game — and ask me to analyse it synoptically: data structures, algorithms,

> networking, security, OS resource use, and database storage. Mark how well I integrated the different areas.

What this helps you practise: Integrating multiple topics around a single real example.

How to use it well: Applications are natural cross-unit hooks. Tracing one across the spec rehearses exactly the linkage examiners reward.

Prompt 88: Reason About Undecidability

Copy this prompt into your AI tool:

> Pose me a question that requires me to reason about what is and isn't computable — for example, whether a problem is equivalent to the halting problem. Mark me on whether I correctly applied the theory rather than guessing intuitively.

What this helps you practise: Building the theory-of-computation thinking that the hardest mark-scheme points reward.

How to use it well: Reduction arguments earn the marks. Practise structuring 'if we could solve this, we could solve the halting problem' arguments.

T

Transform

Learning that isn't reviewed is learning that fades. The Transform stage uses AI to help you reflect on how you studied, log your mistakes, plan your next session, and schedule spaced reviews — turning effort today into memory that lasts to the exam.

Prompt 89: Review the Session

Copy this prompt into your AI tool:

> Summarise what I worked on in this session on [topic]. Tell me what now looks secure, what's still shaky, and what I should revisit — and schedule reviews at widening intervals (1 day, 3 days, 7 days, 14 days) — this is the rhythm the cognitive-science research on spacing shows beats forgetting. Before you schedule the reviews, ask me to compress the single biggest insight from this session into one sentence — the thing I want to remember next year, not just next week. That compression is the learning consolidating.

What this helps you practise: Consolidating a session and planning the right review timing.

How to use it well: Ending each session with a review-and-schedule step is what turns effort into lasting memory.

Prompt 90: Build a Spaced-Repetition Schedule

Copy this prompt into your AI tool:

> Help me build a spaced-repetition schedule for [topic or set of topics] running up to my exam on [date]. Space the reviews so I revisit each topic at widening intervals, and tell me what to do on each review day (definitions Monday, tracing Tuesday, pseudocode Wednesday).

What this helps you practise: Scheduling reviews at the intervals that beat forgetting.

How to use it well: Spacing is one of the most powerful study principles there is. A schedule removes the guesswork and the cramming.

Prompt 91: Build an Error Log

Copy this prompt into your AI tool:

> Help me set up an error log for AP Computer Science. Give me a simple structure to record each mistake (question, my answer, the correct answer, the cause: misread command word, wrong layer, off-by-one, wrong complexity). Then take this mistake I just made — [describe it] — and show me how to log it.

What this helps you practise: Turning mistakes into a targeted, personal revision resource.

How to use it well: Your errors are the most efficient thing you can revise. A good log makes sure you actually learn from each one.

Prompt 92: Plan Tomorrow from Today's Gaps

Copy this prompt into your AI tool:

> Based on the gaps we found in today's session — [list them] — build me a focused plan for tomorrow's revision: what to do, in what order, and how long on each, with the biggest weakness first.

What this helps you practise: Converting today's diagnosis directly into tomorrow's action.

How to use it well: A plan built from real diagnosed gaps beats a generic timetable every time. Let the evidence drive the plan.

Prompt 93: Run a Metacognitive Check

Copy this prompt into your AI tool:

> Ask me reflective questions about how I just studied [topic]: did I trace algorithms by hand, did I write pseudocode without copying, did I check my own code, or did I just re-read? Then suggest one change to my method.

What this helps you practise: Becoming aware of, and improving, how you study.

How to use it well: Studying smarter starts with noticing how you study now. One honest change per week compounds quickly.

Prompt 94: Set Up a Progress Tracker

Copy this prompt into your AI tool:

> Help me create a simple progress tracker for AP Computer Science: a list of all my topics (programming, data structures, algorithms, theory, architecture, OS, networks, databases, ethics, Boolean/numbers) with a confidence rating for each that I can update over time. Suggest how often to revisit and update it.

What this helps you practise: Keeping a clear, evolving picture of where you stand.

How to use it well: A tracker stops you over-revising comfortable topics and neglecting weak ones. Update it honestly after each session.

Prompt 95: Find My Recurring Mistakes

Copy this prompt into your AI tool:

> Here are several mistakes I've made recently: [paste or describe them]. Look for the pattern — is there a recurring type of error (off-by-one, wrong base case, confusing two protocols, misreading a normalisation requirement) — that I keep repeating?

What this helps you practise: Spotting the systematic weakness behind scattered errors.

How to use it well: Individual mistakes are noise; the pattern is signal. Fixing one recurring flaw can lift marks across many questions.

Prompt 96: Plan a Timetable to the Exam

Copy this prompt into your AI tool:

> Work backwards from my exam date [date] to build a realistic AP Computer Science revision timetable across all my topics: [list]. Build in review days, weak-topic time, and full past-paper practice near the end.

What this helps you practise: Turning a daunting syllabus into a paced, finishable plan.

How to use it well: Working back from the exam date keeps the plan realistic. Protect the review and past-paper days — don't let content crowd them out.

Prompt 97: Analyse a Mock Paper

Copy this prompt into your AI tool:

> Here are my results and the questions I dropped marks on in a recent mock: [paste]. Help me analyse it: which topics, which question types (definition, tracing, pseudocode, extended explanation), and which skills cost me marks, and what I should prioritise as a result.

What this helps you practise: Mining a mock for everything it can tell you.

How to use it well: A mock is a diagnostic, not a verdict. The marks you dropped are a precise map of what to revise next.

Prompt 98: Turn Weaknesses into a To-Do List

Copy this prompt into your AI tool:

> Take everything we've identified as weak across my recent sessions — [summarise] — and turn it into a concrete, prioritised to-do list of specific revision actions, not vague intentions (for example: 'do 5 trace tables for recursive algorithms and 5 pseudocode questions on linked lists').

What this helps you practise: Converting fuzzy weaknesses into clear next actions.

How to use it well: 'Revise algorithms' is vague; 'do five quicksort traces and one complexity proof' is an action. Make every item doable.

Prompt 99: Establish a Brain-Dump Routine

Copy this prompt into your AI tool:

> Teach me a brain-dump routine for [topic, e.g. networks]: I write everything I can recall from memory (protocols, layers, addressing, packet headers), you check it against what I should know, and we reflect on what was missing and why. Help me make this a regular habit.

What this helps you practise: Combining free-recall retrieval with reflective review.

How to use it well: The brain dump is pure retrieval; the check-and-reflect is where it sharpens. Doing it regularly builds both memory and self-awareness.

Prompt 100: Consolidate the Week

Copy this prompt into your AI tool:

> It's the end of my study week. I covered [list topics]. Run a consolidation: a quick mixed quiz across all of them, a note of what improved and what's slipping, and a plan for what next week should prioritise.

What this helps you practise: Closing each week with review, reflection, and re-planning.

How to use it well: A weekly consolidation stops topics quietly fading and keeps your plan responsive to how you're actually progressing.

Final Closing Note

You have now worked through 100 prompts designed to help you think more clearly, revise more effectively, and prepare more confidently for your AP Computer Science A exams.

Remember: the goal was never to rely on AI for answers. It was to use AI as a tool to test, challenge, and strengthen your own understanding. The strongest students are not those who avoid difficulty, but those who engage with it deliberately. Each gap you found, each explanation you improved, and each mistake you corrected has strengthened your thinking.

As you continue, aim to depend less on the prompts and more on your own judgement. AI can support you — but your reasoning, clarity, and persistence are what earn the marks.

Approach your exams calmly. Think carefully. Write clearly. You are more prepared than you think.

Using AI Beyond This Book

The prompts in this book are starting points, not final forms. As you grow more confident, begin to modify them:

- Add constraints, for example “limit your answer to three key points”.
- Increase the difficulty gradually as you improve.
- Ask the AI to challenge your reasoning rather than agree with it.
- Request alternative explanations when the first doesn't click.
- Ask it to critique your thinking instead of handing you the answer.

The most powerful use of AI is not asking it to tell you things — it is asking it to test and refine your thinking. Treat AI as a tutor, not a shortcut. The skill of asking better questions will keep mattering long after these exams are over.

About the Velvet Method

The Velvet Method is a study system built by teachers who believe that AI, used well, can give every student the kind of personalised support that was once the preserve of expensive private tutoring.

Our resources are designed around cognitive science — retrieval practice, spaced repetition, schema-building, and metacognition — and around a single principle: AI should strengthen a student's thinking, never replace it.

Explore the full series of subject guides, free revision games and tools, and the Velvet Method course at aistudymethod.co.uk.

The Velvet Method Series

The 100 AI Prompts series supports students across GCSE, A-Level and IBDP. Every title is free to download.

GCSE

- English Language
- English Literature
- Mathematics
- Physics
- Biology
- Chemistry
- Geography
- History
- Computer Science
- Economics
- Business Studies
- Religious Studies
- Psychology
- French
- Spanish
- German

A-Level

- Mathematics
- Further Mathematics
- Physics
- Chemistry
- Biology
- Economics
- History
- Geography
- English Literature
- Psychology
- Computer Science
- Politics
- Business

IBDP

- Mathematics: Analysis & Approaches
- Mathematics: Applications & Interpretation
- Physics
- Chemistry
- Biology
- Economics
- Geography
- History
- English A: Literature
- English A: Language & Literature

- Psychology
- Business Management
- Computer Science

All titles free at aistudymethod.co.uk

The Velvet Method™

AISTUDYMETHOD.CO.UK