GCSE COMPUTER SCIENCE

100 AI PROMPTS

for Smarter Revision *and* Exam Prep

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

by James R. Martin

This book is an independent educational resource and is not affiliated with, endorsed by, or approved by any examination board or awarding organisation.

The author has made use of artificial intelligence tools to assist with drafting, structuring, and generating example material. All educational guidance, explanations, and exam-related advice have been reviewed, edited, and curated by the author. Any resemblance to specific published materials is unintentional.

This book is intended to support revision and exam preparation. It does not replace formal teaching, textbooks, or official specifications. Students are responsible for ensuring that all work submitted for assessment is their own.

ISBN: [TO BE ASSIGNED]

First published 2026

How to Use This Book

For a long time, high-quality tutoring has been a major contributor to elite academic achievement. Used well, AI can now act as a powerful tutor that most students and parents could not previously afford.

This book is a **starting point**, not a rulebook. Each prompt is designed to help you revise, test your understanding, and think more clearly — not to give perfect answers. You are encouraged to **adapt, improve, and remix** these prompts.

You are learning how to think carefully about the questions you ask — a skill that will matter far beyond these exams.

Note on Exam Boards and Syllabi

This collection of 100 AI-powered revision prompts has been designed to support GCSE Computer Science students across all major exam boards, including AQA, Edexcel (Pearson), and OCR. Whether you are following the AQA 8525 specification, the Edexcel 1CP2 course, or the OCR J277 syllabus, these prompts address the core knowledge and skills that every GCSE Computer Science student needs to master.

All three exam boards cover fundamentally similar core content areas: data representation, computer systems and architecture, networking, cyber security, algorithms, programming, and Boolean logic. The prompts in this book are mapped to these shared topics so that no matter which board your school follows, you will find every prompt relevant and useful for your revision.

Where the boards differ most is in their choice of programming language and the specific format of exam questions. AQA uses Python as its reference language, Edexcel allows centres to choose from Python, Java, or C-based languages, and OCR uses a combination of OCR Exam Reference Language and a high-level programming language such as Python. The programming prompts in this book use Python-style pseudocode and concepts that transfer across all these languages, so you can adapt them to whichever language your school teaches.

Each prompt is designed to be used with an AI chatbot to create an interactive, personalised revision experience. You will find prompts that quiz you, walk you through worked examples step by step, challenge you with exam-style questions, and help you identify and fix common misconceptions. The prompts encourage active recall and spaced

repetition, two of the most effective revision strategies supported by cognitive science research.

To get the most from these prompts, work through them alongside your class notes, textbook, and past papers. Use the AI chatbot as a patient tutor that can explain concepts in different ways, generate unlimited practice questions, and give you instant feedback. Remember that while the AI is an excellent revision companion, you should always verify important details against your official specification and mark schemes from your exam board.

Contents

Section 1

Data Representation and Number Systems

Data representation is one of the most fundamental topics in GCSE Computer Science. Every piece of information that a computer processes, whether it is a number, a letter, an image, or a sound, must be represented in binary. Understanding how computers convert between denary (base 10), binary (base 2), and hexadecimal (base 16) number systems is essential, and questions on these conversions appear on every exam paper.

Beyond simple number conversions, you need to understand how binary arithmetic works, including binary addition and the concept of overflow. You also need to know how characters are encoded using systems like ASCII and Unicode, how images are represented using pixels with properties such as resolution and colour depth, and how sound is digitised using sample rate and bit depth. These topics connect directly to questions about file sizes and data storage.

Compression is another key area within data representation. You must be able to explain the difference between lossy and lossless compression, give examples of when each would be appropriate, and understand the trade-offs between file size and quality. This section will help you build fluency in all of these areas through interactive practice with an AI tutor.

Prompt 1: Binary to Denary Conversion Drill

Copy this prompt into your AI tool:

Give me a GCSE-style question about Computer Science. Give me 5 binary numbers (8-bit) one at a

time and ask me to convert each one to denary. After I answer each one, tell me if I am correct and show the working using place values (128, 64, 32, 16, 8, 4, 2, 1). If I get one wrong, explain where my mistake was before moving to the next question. After all 5, give me a score out of 5 and suggest what to practise if I dropped marks.

What this helps you practise:

Converting 8-bit binary numbers to denary using place values.

How to use it well:

Work through all 5 conversions, writing your working on paper before typing your answer. This builds the speed you need in the exam.

Prompt 2: Denary to Binary Conversion Practice

Copy this prompt into your AI tool:

Act as my GCSE Computer Science tutor. Give me 5 denary numbers between 0 and 255, one at a time, and ask me to convert each to an 8-bit binary number. After each answer, check my work and show the correct method using repeated division by 2 or the place value method. Explain any errors I make. After all 5, tell me my score and offer to generate 5 harder examples if I scored well.

What this helps you practise:

Converting denary numbers to 8-bit binary representations.

How to use it well:

Try the place value method first, then verify using division by 2. Practise both methods so you can use whichever feels faster in the exam.

Prompt 3: Hexadecimal Conversion Challenge

Copy this prompt into your AI tool:

You are a GCSE Computer Science examiner. Test me on hexadecimal conversions in three rounds.

Round 1: give me 3 hexadecimal values to convert to denary. Round 2: give me 3 denary values to convert to hexadecimal. Round 3: give me 3 binary values to convert to hexadecimal by splitting into nibbles. Present each question one at a time, mark my answer, and show the correct working. Finish with a summary of which conversion direction I need more practice on.

What this helps you practise:

Converting between hexadecimal, denary, and binary number systems.

How to use it well:

Remember that each hex digit maps to exactly one nibble (4 bits). Write out the hex digit values 0-F alongside their denary equivalents before starting.

Prompt 4: Why Hexadecimal Matters

Copy this prompt into your AI tool:

Act as my GCSE Computer Science tutor. Explain to me why hexadecimal is used in computer science instead of just using binary or denary. Give me three specific real-world examples of where hexadecimal is used (such as colour codes, MAC addresses, or error codes). Then quiz me with 3 questions: ask me to explain one advantage of hex over binary, one advantage over denary, and to give my own example of where hex is used. Give feedback on each answer.

What this helps you practise:

Understanding the purpose and advantages of hexadecimal representation.

How to use it well:

Focus on understanding why hex is a convenient shorthand for binary, not just how to convert between them.

Prompt 5: Binary Addition and Overflow

Copy this prompt into your AI tool:

Ask me to demonstrate my Computer Science knowledge. Teach me binary addition step by step. Start by reminding me of the four binary addition rules (0+0, 0+1, 1+0, 1+1) and what a carry bit is. Then give me 5 binary addition problems using 8-bit numbers, one at a time. For at least two of them, make the result cause an overflow. After each answer, show the correct column-by-column working. When overflow occurs, explain what overflow means, why it happens, and what problems it causes in computing.

What this helps you practise:

Performing binary addition and identifying overflow errors in 8-bit systems.

How to use it well:

Line up the bits carefully in columns when adding. Pay special attention to the carry bit and watch for overflow when the result exceeds 8 bits.

Prompt 6: Binary Shifts Explained

Copy this prompt into your AI tool:

Act as my GCSE Computer Science tutor. Explain what a binary left shift and a binary right shift do to a binary number, and what effect each has on the denary value. Give me 4 worked examples: two left shifts and two right shifts. Then test me with 4 questions where I have to perform the shift and state the new denary value. After each answer, confirm if I am right and explain any errors. Also ask me what data can be lost during a right shift.

What this helps you practise:

Performing binary left and right shifts and understanding their effect on values.

How to use it well:

Remember that a left shift by one place multiplies the denary value by 2, and a right shift divides by 2. Watch for bits falling off the end.

Prompt 7: ASCII and Unicode Character Encoding

Copy this prompt into your AI tool:

Set me a challenge: Explain the difference between ASCII and Unicode character encoding. Cover how many bits each uses, how many characters each can represent, and why Unicode was developed. Then give me a short quiz: ask me 5 questions mixing true/false and short answer, covering topics like the number of characters in the ASCII set, why Unicode uses more storage, and whether Unicode is backwards compatible with ASCII. Mark each answer and correct any misconceptions.

What this helps you practise:

Understanding ASCII and Unicode character encoding systems and their differences.

How to use it well:

Focus on the reasons behind Unicode's development (international characters, emojis) and the trade-off between character range and file size.

Prompt 8: Image Representation Deep Dive

Copy this prompt into your AI tool:

Act as my GCSE Computer Science tutor. Explain how images are represented in binary, covering the terms pixel, resolution, colour depth, and metadata. Show me the formula for calculating image file size (width x height x colour depth). Then give me 4 calculation questions where I work out the file size of an image given its dimensions and colour depth. Present them one at a time, check my answers, and show the working. Include one question where I have to convert the result from bits to kilobytes.

What this helps you practise:

Understanding image representation and calculating image file sizes from resolution and colour depth.

How to use it well:

Memorise the file size formula and practise converting between bits, bytes, kilobytes, and megabytes. These calculations come up frequently in exams.

Prompt 9: The Effect of Resolution and Colour Depth

Copy this prompt into your AI tool:

You are a GCSE Computer Science examiner. Present me with a scenario: an image editor lets me change the resolution and colour depth of a photograph. Ask me to predict and explain what happens to the image quality and file size when I: (1) increase the resolution, (2) decrease the colour depth, (3) increase both, (4) decrease both. After each prediction, give me feedback and correct any misunderstandings. Then ask me a 6-mark exam-style question comparing two images with different resolutions and colour depths.

What this helps you practise:

Explaining the trade-offs between image quality and file size when changing resolution and colour depth.

How to use it well:

Think about what each change does independently first, then combine the effects. Use precise terminology in your answers.

Prompt 10: Sound Representation and Sampling

Copy this prompt into your AI tool:

Present me with a Computer Science challenge. Explain how sound is represented digitally, covering the terms sample rate, bit depth, and bit rate. Explain how each of these affects sound quality and file size. Then quiz me with 5 questions: include calculating the file size of an audio clip given its

sample rate, bit depth, and duration; explaining what happens to quality when sample rate is increased; and defining bit rate. Mark each answer and provide detailed feedback.

What this helps you practise:

Understanding sound digitisation including sample rate, bit depth, duration, and file size calculations.

How to use it well:

Learn the formula: file size = sample rate x bit depth x duration (x number of channels for stereo). Practise unit conversions as answers often need to be in KB or MB.

Prompt 11: Lossy vs Lossless Compression

Copy this prompt into your AI tool:

Set me a challenge: Explain the difference between lossy and lossless compression. For each type, tell me how it works, give two real-world examples of file formats that use it, and explain when it would be the best choice. Then present me with 5 scenarios (such as compressing a medical image, streaming music, archiving important documents, sending photos on social media, and compressing a program file) and ask me to decide whether lossy or lossless compression is more appropriate for each, with a justification. Give feedback on each answer.

What this helps you practise:

Distinguishing between lossy and lossless compression and choosing the appropriate type for different contexts.

How to use it well:

For each scenario, think about whether the data can afford to lose quality or whether every bit of original data must be preserved.

Prompt 12: Run-Length Encoding (RLE)

Copy this prompt into your AI tool:

Test me on this Computer Science topic. Explain how run-length encoding works as a form of lossless compression. Show me a worked example of encoding a simple sequence of data using RLE. Then give me 3 sequences to encode and 2 encoded sequences to decode, one at a time. After each answer, show the correct result and explain any errors. Finally, ask me to explain one scenario where RLE would be very effective and one where it would not work well.

What this helps you practise:

Applying run-length encoding to compress and decompress data sequences.

How to use it well:

Look for repeated consecutive values in the data. RLE works best when there are long runs of the same value, such as in simple graphics.

Prompt 13: Units of Data Storage

Copy this prompt into your AI tool:

You are a GCSE Computer Science examiner. Teach me the units of data storage from bits to petabytes, including the relationship between each unit (bit, nibble, byte, kilobyte, megabyte, gigabyte, terabyte, petabyte). Then test me with 8 quick-fire conversion questions, such as converting bytes to bits, kilobytes to bytes, and megabytes to kilobytes. Present them one at a time and mark each answer immediately. Finish by asking me why understanding these units matters for file size calculations.

What this helps you practise:

Converting between units of data storage and understanding their relationships.

How to use it well:

Create a reference table of units and their multipliers. Being quick with these conversions saves time in calculation questions.

Prompt 14: Data Representation Exam Buster

Copy this prompt into your AI tool:

Test me on this Computer Science topic. Give me a mini exam on data representation with 6 mixed questions covering: one binary-to-hex conversion, one image file size calculation, one question about the difference between ASCII and Unicode, one binary addition with overflow detection, one question about sound sampling, and one comparing lossy and lossless compression. Present each question one at a time, wait for my answer, then mark it and give the model answer. At the end, give me an overall score and identify my weakest area with a suggestion for what to revise next.

What this helps you practise:

Applying data representation knowledge across all sub-topics in exam conditions.

How to use it well:

Treat this like a real exam. Time yourself, write your answers on paper first, and then type them in to be marked.

Section 2

Computer Systems and Architecture

Understanding how a computer works at the hardware level is a core requirement of GCSE Computer Science. You need to know the key components of a CPU, including the arithmetic logic unit (ALU), the control unit (CU), the registers, and cache memory. You must also be able to describe the Von Neumann architecture and explain how the fetch-decode-execute cycle allows the CPU to process instructions sequentially.

This section also covers the differences between primary storage (RAM and ROM) and secondary storage technologies including magnetic, optical, and solid-state devices. You should be able to compare these storage types in terms of speed, capacity, cost, durability, and portability. Understanding embedded systems and how they differ from general-purpose computers is another important topic that frequently appears in exams.

Operating systems and utility software round out this section. You need to know the key functions of an operating system, such as memory management, process scheduling, and managing peripherals, as well as the role of utility software like defragmentation tools, encryption software, and backup utilities. These prompts will help you build a thorough understanding of all these concepts through structured, interactive revision.

Prompt 15: Von Neumann Architecture Explained

Copy this prompt into your AI tool:

Present me with a Computer Science challenge. Explain the Von Neumann architecture to me,

including the four main components: the CPU, memory, input devices, and output devices. Describe how they are connected by buses (data bus, address bus, control bus). Then quiz me: ask me to name and describe each bus, explain what is meant by the stored program concept, and draw a labelled diagram in text form. Give feedback on each answer and correct any mistakes.

What this helps you practise:

Understanding and describing the Von Neumann architecture and the stored program concept.

How to use it well:

Sketch the architecture diagram from memory before starting. This is a common exam question and you need to be able to reproduce it quickly.

Prompt 16: CPU Components Breakdown

Copy this prompt into your AI tool:

Test me on this Computer Science topic. Teach me about the key components inside the CPU: the arithmetic logic unit (ALU), the control unit (CU), the cache, and the registers (including the program counter, memory address register, memory data register, and accumulator). Explain the role of each component. Then test me with 6 questions where you describe what a component does and I have to name it, or you name a component and I have to describe its function. Mark each answer and provide corrections.

What this helps you practise:

Identifying and explaining the function of each CPU component and register.

How to use it well:

Create flashcards for each component after this session. Being able to recall these quickly is essential for short-answer exam questions.

Prompt 17: The Fetch-Decode-Execute Cycle

Copy this prompt into your AI tool:

Give me a GCSE-style question about Computer Science. Walk me through the fetch-decode-execute cycle step by step, explaining what happens at each stage and which CPU components and registers are involved. Then give me a jumbled list of 8 steps from the cycle and ask me to put them in the correct order. After I answer, show the correct order and explain any mistakes. Finally, ask me a 4-mark exam question about the cycle and mark my response against a model answer.

What this helps you practise:

Describing the fetch-decode-execute cycle and the role of each register within it.

How to use it well:

Trace through the cycle with a specific example instruction to help make it concrete. Think about what the program counter, MAR, and MDR contain at each step.

Prompt 18: Factors Affecting CPU Performance

Copy this prompt into your AI tool:

Ask me to demonstrate my Computer Science knowledge. Explain the three main factors that affect CPU performance: clock speed, number of cores, and cache size. For each factor, explain what it is, how it improves performance, and any limitations or trade-offs. Then give me 4 scenario-based questions, such as asking which factor would most improve performance for a specific type of task (gaming, video editing, running multiple programs). Mark each answer and explain the reasoning.

What this helps you practise:

Explaining and comparing the factors that affect CPU performance.

How to use it well:

Learn a clear definition and one advantage and one limitation for each factor. Exam questions often ask you to explain rather than just name these factors.

Prompt 19: RAM vs ROM Showdown

Copy this prompt into your AI tool:

Give me a GCSE-style question about Computer Science. Create an interactive comparison between RAM and ROM. Start by asking me to tell you what I already know about each. Then fill in any gaps, covering volatility, read/write capability, purpose, and typical usage. Present me with a table to complete (in text format) with rows for each property. After I fill it in, correct any errors. Then give me 3 exam-style questions comparing RAM and ROM, mark my answers, and explain the marking criteria.

What this helps you practise:

Comparing RAM and ROM in terms of volatility, function, and usage.

How to use it well:

The most common mistake is confusing volatility with whether data can be written. Make sure you are clear on both properties.

Prompt 20: Secondary Storage Comparison

Copy this prompt into your AI tool:

Ask me to demonstrate my Computer Science knowledge. Explain the three types of secondary storage: magnetic (hard disk drives), optical (CDs, DVDs, Blu-ray), and solid state (SSDs, USB flash drives). For each type, cover how it stores data, its advantages, and its disadvantages in terms of speed, capacity, cost, durability, and portability. Then give me 5 scenario questions where I choose the most

suitable storage type and justify my choice. Mark each answer with feedback.

What this helps you practise:

Comparing magnetic, optical, and solid-state secondary storage technologies.

How to use it well:

Make a comparison table covering all five properties for each storage type. Exam questions often ask you to recommend a storage type for a given scenario.

Prompt 21: Why Do We Need Secondary Storage?

Copy this prompt into your AI tool:

Present me with a Computer Science challenge. Ask me to explain why secondary storage is needed in a computer system, given that we already have RAM. Guide me to include the concepts of volatility, capacity, and cost in my explanation. Then ask me to compare virtual memory with physical RAM, explaining what virtual memory is and why it is used. Mark my answers and provide a model response for a 4-mark exam question on this topic.

What this helps you practise:

Explaining the need for secondary storage and the concept of virtual memory.

How to use it well:

Think about what happens when you turn the computer off (volatility) and when RAM runs out (virtual memory). These are key exam points.

Prompt 22: Embedded Systems in the Real World

Copy this prompt into your AI tool:

Set me a challenge: Explain what an embedded system is and how it differs from a general-purpose computer. Give me 5 examples of embedded systems in everyday life and explain what makes each one an

embedded system. Then quiz me with 4 questions: ask me to identify whether a described device is an embedded system or a general-purpose computer, and to justify my answer. Mark each response and explain any errors.

What this helps you practise:

Identifying and describing embedded systems and how they differ from general-purpose computers.

How to use it well:

Remember the key features: single dedicated function, built into a larger device, limited or no user interface. These are the points examiners look for.

Prompt 23: Operating System Functions

Copy this prompt into your AI tool:

Ask me to demonstrate my Computer Science knowledge. Explain the main functions of an operating system, covering: memory management, process management and multitasking, file management, peripheral management and device drivers, user interface provision, and security (user accounts and access control). For each function, give a brief real-world example. Then test me with 6 questions where you describe a situation and I identify which OS function is being used. Mark each answer.

What this helps you practise:

Describing the key functions of an operating system with examples.

How to use it well:

Think about what happens behind the scenes every time you use your computer. Each action you take involves multiple OS functions working together.

Prompt 24: Utility Software Explained

Copy this prompt into your AI tool:

Act as my GCSE Computer Science tutor. Explain what utility software is and describe the purpose of these common utility programs: defragmentation tools, compression software, backup utilities, encryption software, and antivirus/anti-malware tools. For each, explain why it is needed and when a user would use it. Then quiz me: describe 5 scenarios and ask me to name the utility software that would help. Mark each answer and correct any mistakes.

What this helps you practise:

Identifying and explaining the purpose of common utility software.

How to use it well:

Learn one clear sentence describing the purpose of each utility. Exam questions often ask you to describe rather than simply name them.

Prompt 25: Memory Hierarchy Challenge

Copy this prompt into your AI tool:

Test me on this Computer Science topic. Explain the memory hierarchy from fastest to slowest: registers, cache, RAM, secondary storage. For each level, explain its speed, capacity, cost per byte, and volatility. Ask me to arrange the levels in order of speed and then in order of capacity. Then give me 4 exam-style questions about why the hierarchy exists, why we cannot just use the fastest memory for everything, and how cache improves CPU performance. Mark each answer with detailed feedback.

What this helps you practise:

Understanding the memory hierarchy and the trade-offs between speed, capacity, and cost.

How to use it well:

Remember the pattern: as you move down the

hierarchy, speed decreases but capacity increases and cost per byte decreases.

Prompt 26: Systems Architecture Diagram Quiz

Copy this prompt into your AI tool:

Act as my GCSE Computer Science tutor. Describe a simplified CPU architecture diagram in text and ask me to label the components: ALU, CU, cache, registers, buses (data, address, control), RAM, and I/O devices. Give me the diagram with some labels missing and ask me to fill them in. After I answer, show the fully labelled version and explain any corrections. Then ask me a 6-mark question about how the CPU uses these components to execute an instruction.

What this helps you practise:

Labelling and explaining CPU architecture diagrams.

How to use it well:

Practise drawing and labelling this diagram from memory. It is one of the most commonly tested diagrams in GCSE Computer Science.

Prompt 27: Cloud vs Local Storage

Copy this prompt into your AI tool:

Test me on this Computer Science topic. Explain the differences between cloud storage and local storage. Cover the advantages and disadvantages of each in terms of accessibility, security, cost, reliability, and internet dependency. Then present me with 4 scenario-based questions where I advise a user on whether cloud or local storage is more appropriate. Mark each answer and check that I have given balanced reasoning.

What this helps you practise:

Comparing cloud storage and local storage for different use cases.

How to use it well:

Exam questions on this topic often ask for both advantages and disadvantages. Always consider both sides even if the question seems to favour one.

Prompt 28: Computer Systems Rapid Recall

Copy this prompt into your AI tool:

You are a GCSE Computer Science examiner. Give me a rapid-fire quiz of 10 short-answer questions on computer systems and architecture. Cover: naming CPU components, stating the purpose of a register, explaining the fetch-decode-execute cycle in one sentence, comparing RAM and ROM, identifying storage types, and naming OS functions. Give me each question one at a time, allow me 30 seconds of thinking time, then mark my answer. At the end, give me a score out of 10 and identify my two weakest areas.

What this helps you practise:

Quick recall of key computer systems and architecture facts under timed conditions.

How to use it well:

Use this as a warm-up quiz before moving on to deeper revision. Speed of recall matters in the exam where time is limited.

Section 3

Networks and Cyber Security

Networking is a vital topic in GCSE Computer Science, covering everything from the basic types of networks to the protocols that make the internet work. You need to understand the difference between a LAN (local area network) and a WAN (wide area network), be able to describe common network topologies such as star and mesh, and explain the role of hardware components including routers, switches, wireless access points, and network interface cards (NICs).

Protocols are the agreed-upon rules that allow devices to communicate over a network. You should be familiar with the TCP/IP protocol stack, HTTP and HTTPS for web browsing, FTP for file transfer, and email protocols. Understanding how data is transmitted using packet switching, including how packets are routed, sequenced, and reassembled, is another frequently examined topic.

Cyber security is increasingly important in both the real world and in GCSE exams. You must be able to describe different types of malware (viruses, worms, trojans, ransomware, spyware), explain social engineering attacks such as phishing and shoulder surfing, and discuss prevention methods including firewalls, encryption, authentication, and network policies. These prompts will test your knowledge across all of these areas.

Prompt 29: LAN vs WAN Essentials

Copy this prompt into your AI tool:

Quiz me on GCSE Computer Science. Explain the difference between a LAN and a WAN, covering geographical size, ownership, hardware used, and

typical data transfer speeds. Give me two real-world examples of each. Then test me with 5 questions: some asking me to classify a described network as LAN or WAN, and some asking me to state characteristics of each type. Mark each answer and correct any misconceptions.

What this helps you practise:

Distinguishing between LANs and WANs and describing their key characteristics.

How to use it well:

Focus on the four key differences: size, ownership, connection media, and speed. These are the comparison points examiners expect.

Prompt 30: Network Topologies Compared

Copy this prompt into your AI tool:

You are a GCSE Computer Science examiner. Explain the star topology and mesh topology for computer networks. For each, describe how devices are connected, draw the layout in text form, and list the advantages and disadvantages. Then give me 4 exam-style questions: ask me to recommend a topology for a given scenario, explain an advantage of one over the other, and describe what happens if a connection fails in each topology. Mark each answer with feedback.

What this helps you practise:

Comparing star and mesh network topologies, including their advantages and disadvantages.

How to use it well:

Think about single points of failure, cost, and scalability when comparing topologies. These are the evaluation points examiners look for.

Prompt 31: Network Hardware Identification

Copy this prompt into your AI tool:

Quiz me on GCSE Computer Science. Describe the function of each of these network hardware components: router, switch, hub, wireless access point (WAP), network interface card (NIC), and transmission media (copper, fibre optic, wireless). Then give me a matching exercise where you list 6 descriptions and I match each to the correct hardware component. After I answer, mark my matches and explain any errors. Finally, ask me why a switch is preferred over a hub in modern networks.

What this helps you practise:

Identifying network hardware components and explaining their functions.

How to use it well:

The key distinction between a switch and a hub is a favourite exam question. Make sure you can explain why switches are more efficient.

Prompt 32: TCP/IP Protocol Stack

Copy this prompt into your AI tool:

Set me a challenge: Explain the four layers of the TCP/IP protocol stack: application layer, transport layer, internet layer, and network interface layer. For each layer, describe its function and name a protocol that operates at that layer. Then test me: give me 5 questions where you name a protocol or describe a function, and I identify the correct layer. Mark each answer and provide explanations for any I get wrong.

What this helps you practise:

Understanding the TCP/IP protocol stack layers and their associated protocols.

How to use it well:

Learn the layers in order and associate at least one protocol with each layer. Think about what each layer adds to the data as it passes through.

Prompt 33: Packet Switching Step by Step

Copy this prompt into your AI tool:

Give me a GCSE-style question about Computer Science. Explain how packet switching works, covering: how data is broken into packets, what information is included in a packet header (source address, destination address, sequence number, packet size), how packets are routed across the network, and how they are reassembled at the destination. Then ask me 4 questions: why packets may take different routes, what happens if a packet is lost, why sequence numbers are needed, and one advantage of packet switching over circuit switching. Mark each answer.

What this helps you practise:

Describing the process of packet switching and the structure of data packets.

How to use it well:

Trace the journey of a single packet from source to destination. Understanding this journey helps you answer extended response questions about data transmission.

Prompt 34: Common Network Protocols

Copy this prompt into your AI tool:

Test me on this Computer Science topic. Explain what a network protocol is and why protocols are needed. Then describe the purpose of each of these protocols: HTTP, HTTPS, FTP, SMTP, IMAP, TCP, IP, and UDP. Test me with a quiz: give me 8 scenarios (such as loading a web page, sending an email, downloading a file, streaming video) and ask me to identify which protocol or protocols would be used. Mark each answer and explain the reasoning.

What this helps you practise:

Identifying common network protocols and understanding when each is used.

How to use it well:

Group the protocols by function (web, email, file transfer, transport) to help remember them. Know at least one sentence describing each protocol's purpose.

Prompt 35: Types of Malware

Copy this prompt into your AI tool:

Ask me to demonstrate my Computer Science knowledge. Explain each of these types of malware: virus, worm, trojan, ransomware, spyware, and adware. For each, describe how it works, how it spreads, and what damage it causes. Then test me with 6 scenarios describing a cyber attack and ask me to identify the type of malware involved. Mark each answer and explain the distinguishing features I should look for.

What this helps you practise:

Identifying and describing different types of malware and how they operate.

How to use it well:

Learn the key distinguishing feature of each malware type: viruses attach to files, worms self-replicate across networks, trojans disguise themselves, and so on.

Prompt 36: Social Engineering Attacks

Copy this prompt into your AI tool:

Test me on this Computer Science topic. Explain what social engineering means in the context of cyber security. Describe these social engineering techniques: phishing, spear phishing, shoulder surfing, pharming, and pretexting. For each, give an example of how it might be carried out. Then quiz

me with 5 scenarios and ask me to identify the social engineering technique being used and suggest one way the victim could have protected themselves. Mark each answer.

What this helps you practise:

Recognising social engineering attack methods and suggesting appropriate countermeasures.

How to use it well:

Social engineering targets people rather than technology. Think about what human weakness each technique exploits.

Prompt 37: Encryption and Authentication

Copy this prompt into your AI tool:

Set me a challenge: Explain the concepts of encryption and authentication in cyber security. Cover what encryption is, the difference between symmetric and asymmetric encryption at a basic level, and why HTTPS uses encryption. Then explain authentication methods including passwords, two-factor authentication, and biometrics. Quiz me with 5 questions covering: when encryption should be used, advantages and disadvantages of biometric authentication, and what makes a strong password. Mark each answer.

What this helps you practise:

Understanding encryption and authentication methods and their importance in cyber security.

How to use it well:

Focus on being able to explain why encryption is needed (to prevent interception) and why authentication is needed (to verify identity). These are distinct concepts.

Prompt 38: Firewalls and Network Security

Copy this prompt into your AI tool:

Give me a GCSE-style question about Computer Science. Explain what a firewall is and how it helps protect a network. Describe how firewalls can filter traffic based on rules, ports, and IP addresses. Then broaden the discussion to other network security measures: MAC address filtering, access control lists, penetration testing, and network policies. Quiz me with 4 scenario-based questions where I recommend appropriate security measures for a described network. Mark my answers and check I have justified my choices.

What this helps you practise:

Explaining firewall functionality and selecting appropriate network security measures.

How to use it well:

Remember that security works in layers. Exam answers that mention multiple security measures working together tend to score higher marks.

Prompt 39: Network Vulnerabilities and Prevention

Copy this prompt into your AI tool:

Quiz me on GCSE Computer Science. Explain these common network vulnerabilities: SQL injection, brute force attacks, denial of service (DoS) attacks, man-in-the-middle attacks, and unpatched software. For each, explain how the attack works and one method of prevention. Then present 5 short scenarios describing a security incident and ask me to identify the vulnerability being exploited and recommend a prevention method. Mark each answer with detailed feedback.

What this helps you practise:

Identifying network vulnerabilities and recommending appropriate prevention strategies.

How to use it well:

For each vulnerability, learn the attack method and

at least one matching prevention technique. Exam questions often ask for both.

Prompt 40: Networks and Security Exam Practice

Copy this prompt into your AI tool:

Give me a GCSE-style question about Computer Science. Give me a mini exam on networks and cyber security with 6 questions: one about LAN vs WAN, one about a network topology, one about protocols, one about packet switching, one about malware, and one 6-mark question about how an organisation can protect its network from cyber threats. Present each one at a time, mark my answer, and give the model answer. At the end, give me a score and highlight which topic I should revise further.

What this helps you practise:

Applying networks and cyber security knowledge across all sub-topics in exam conditions.

How to use it well:

Attempt the 6-mark question using the structure: point, explanation, example for each security measure. This is how examiners expect extended answers to be structured.

Section 4

Algorithms and Computational Thinking

Algorithms are at the heart of computer science. An algorithm is a step-by-step set of instructions to solve a problem, and understanding how to design, follow, and evaluate algorithms is essential for your GCSE. You need to be comfortable with the three key principles of computational thinking: decomposition (breaking a problem into smaller parts), abstraction (removing unnecessary detail), and algorithmic thinking (creating step-by-step solutions).

You will be expected to read, write, and trace algorithms presented as pseudocode or flowcharts. This means understanding standard pseudocode conventions, being able to follow the flow of execution through a program, and using trace tables to track variable values as an algorithm runs. These skills are tested directly in the exam and are essential for debugging programs.

Searching and sorting algorithms are a major component of this topic. You need to know how linear search and binary search work, when each is appropriate, and how their efficiency compares. For sorting, you must understand bubble sort and merge sort (and insertion sort for some boards), be able to trace through them, and compare their efficiency. These prompts will build your confidence in all of these areas.

Prompt 41: Decomposition and Abstraction Explained

Copy this prompt into your AI tool:

Present me with a Computer Science challenge. Explain the concepts of decomposition and

abstraction in computational thinking. Give me two real-world examples of each. Then present me with a complex problem (such as creating a school registration system) and ask me to demonstrate decomposition by breaking it into sub-problems, and abstraction by identifying which details are essential and which can be ignored. Give feedback on my answer and suggest improvements.

What this helps you practise:

Applying decomposition and abstraction to solve problems.

How to use it well:

Practise decomposition by taking any large task and breaking it into the smallest possible steps. For abstraction, think about what a model needs to include and what it can leave out.

Prompt 42: Pseudocode Reading Challenge

Copy this prompt into your AI tool:

Give me a GCSE-style question about Computer Science. Show me 4 short pseudocode algorithms one at a time. For each, ask me to read the code and describe what the algorithm does, what output it produces for a given input, and to identify the programming constructs used (sequence, selection, iteration). After each answer, confirm the correct output and explanation. If I make errors, walk me through the code line by line. Increase the difficulty with each example.

What this helps you practise:

Reading and interpreting pseudocode algorithms to determine their purpose and output.

How to use it well:

Use a trace table even for simple programs. Writing down variable values at each step prevents mistakes and is a skill the exam tests directly.

Prompt 43: Flowchart Interpretation

Copy this prompt into your AI tool:

You are a GCSE Computer Science examiner. Describe 3 flowcharts to me using text descriptions of the shapes and flow (since we cannot draw images). For each, use the correct flowchart symbols: ovals for start/end, rectangles for processes, diamonds for decisions, and parallelograms for input/output. Ask me to trace through each flowchart with given inputs and state the output. After each, confirm the correct answer and explain the logic flow. Then ask me to identify an error in a fourth flowchart that contains a deliberate bug.

What this helps you practise:

Tracing through flowcharts and understanding standard flowchart symbols.

How to use it well:

Memorise the four standard flowchart symbols and what each represents. Follow the arrows carefully, especially at decision points where the flow branches.

Prompt 44: Linear Search Step by Step

Copy this prompt into your AI tool:

Give me a GCSE-style question about Computer Science. Explain how the linear search algorithm works, write it out in pseudocode, and walk me through an example with a list of 8 unsorted numbers searching for a specific value. Then ask me to trace through a linear search on a different list, recording each comparison made. After I complete the trace, check my work. Then ask me to state the best-case, worst-case, and average number of comparisons for a linear search on a list of n items.

What this helps you practise:
Understanding, tracing, and analysing the linear search algorithm.
How to use it well:
Linear search is simple but inefficient for large datasets. Understand why it works on unsorted data but binary search does not.

Prompt 45: Binary Search Step by Step
Copy this prompt into your AI tool:
Test me on this Computer Science topic. Explain how the binary search algorithm works, emphasising that the data must be sorted first. Write the algorithm in pseudocode and trace through an example with a sorted list of 15 items. Then give me a sorted list of 10 items and a target value, and ask me to perform a binary search, showing the midpoint, which half is discarded, and the new sublist at each step. Check my trace and correct any errors. Then ask me to compare binary search with linear search in terms of efficiency.
What this helps you practise:
Understanding, tracing, and comparing the binary search algorithm with linear search.
How to use it well:
Calculate the midpoint at each step and remember to round down if necessary. The key advantage of binary search is that it eliminates half the data with each comparison.

Prompt 46: Bubble Sort Walkthrough
Copy this prompt into your AI tool:
Quiz me on GCSE Computer Science. Explain how the bubble sort algorithm works, including the concept of passes, comparisons, and swaps. Write the algorithm in pseudocode. Then give me a list of 6 unsorted numbers and ask me to perform a complete

bubble sort, showing the state of the list after each pass. Check my work after each pass. After I finish, ask me to state how many passes and comparisons were needed, and explain when we know the sort is complete.

What this helps you practise:

Tracing through the bubble sort algorithm and understanding its efficiency.

How to use it well:

Track each comparison and swap carefully. The optimised version stops early if no swaps occur in a pass, which shows the list is already sorted.

Prompt 47: Merge Sort Explained

Copy this prompt into your AI tool:

Ask me to demonstrate my Computer Science knowledge. Explain how the merge sort algorithm works using the divide-and-conquer approach. Show me the splitting phase and the merging phase with a worked example using a list of 8 numbers. Then give me a different list of 8 numbers and ask me to show the splitting and merging steps. Check each step of my answer. Then ask me to compare merge sort with bubble sort in terms of efficiency, memory usage, and suitability for large datasets.

What this helps you practise:

Understanding and tracing the merge sort algorithm, and comparing it with bubble sort.

How to use it well:

Draw out the tree diagram showing the splits and merges. Merge sort is more efficient than bubble sort for large lists but uses more memory.

Prompt 48: Insertion Sort Practice

Copy this prompt into your AI tool:

Present me with a Computer Science challenge. Explain how the insertion sort algorithm works by

maintaining a sorted portion and inserting each new element into its correct position. Write the algorithm in pseudocode and trace through an example with 6 numbers. Then give me a new list of 7 numbers and ask me to perform an insertion sort, showing the sorted portion after each insertion. Check my work and correct any errors. Ask me when insertion sort performs well compared to other sorting algorithms.

What this helps you practise:

Tracing through the insertion sort algorithm and understanding when it is efficient.

How to use it well:

Think of insertion sort like sorting a hand of playing cards: you pick up each card and insert it into the correct position among the cards you have already sorted.

Prompt 49: Trace Tables Mastery

Copy this prompt into your AI tool:

Ask me to demonstrate my Computer Science knowledge. Explain what a trace table is and why it is used to test algorithms. Show me a worked example with a simple loop-based algorithm. Then give me 3 pseudocode algorithms of increasing difficulty and ask me to complete a trace table for each, tracking all variables and any output. After each one, show me the correct trace table and highlight any differences. Explain common mistakes to avoid when completing trace tables.

What this helps you practise:

Completing trace tables accurately to track variable values and output through an algorithm.

How to use it well:

Create a column for every variable and one for output. Update values systematically line by line. This is one of the most common exam question types.

Prompt 50: Algorithm Efficiency and Complexity

Copy this prompt into your AI tool:

Quiz me on GCSE Computer Science. Explain what algorithm efficiency means at GCSE level, focusing on the number of steps or comparisons required as the input size grows. Compare the efficiency of: linear search vs binary search, and bubble sort vs merge sort. Use specific examples with lists of 10, 100, and 1000 items to illustrate how the number of operations differs. Then quiz me with 4 questions about which algorithm would be faster in different scenarios and why. Mark each answer.

What this helps you practise:

Understanding and comparing algorithm efficiency for searching and sorting algorithms.

How to use it well:

You do not need to know Big O notation for GCSE, but you should be able to explain in words why some algorithms take many more steps than others as the data grows.

Prompt 51: Writing Pseudocode from Scratch

Copy this prompt into your AI tool:

Ask me to demonstrate my Computer Science knowledge. Give me 3 problems of increasing difficulty and ask me to write a pseudocode solution for each. Problem 1: a program that asks for a number and states whether it is even or odd. Problem 2: a program that finds the largest value in a list of numbers. Problem 3: a program that validates a password by checking it is at least 8 characters long and contains a digit. After each, compare my pseudocode with a model answer, point out any logical errors, and suggest improvements.

What this helps you practise:

Writing clear, correct pseudocode algorithms from problem descriptions.

How to use it well:

Start by identifying the inputs, processes, and outputs before writing any code. Use indentation to show the structure of your selection and iteration.

Prompt 52: Searching and Sorting Algorithm Selector

Copy this prompt into your AI tool:

Present me with a Computer Science challenge. Present me with 6 scenarios involving data that needs to be searched or sorted. For each scenario, tell me the size of the data, whether it is already sorted, and any constraints (such as limited memory or need for speed). Ask me to choose the most appropriate searching or sorting algorithm and justify my choice. After each answer, evaluate my reasoning and provide the best answer with explanation. Summarise the key decision factors at the end.

What this helps you practise:

Selecting the most appropriate search or sort algorithm based on the characteristics of the data and the requirements.

How to use it well:

Consider whether the data is sorted (needed for binary search), how large the dataset is (merge sort for large data), and memory constraints (merge sort uses extra memory).

Section 5

Programming Concepts

Programming is a core component of GCSE Computer Science, and you will be tested on your ability to read, write, and debug code. While the specific programming language varies by exam board, the underlying concepts are universal: variables, constants, data types, operators, selection, iteration, arrays and lists, subroutines, and file handling. These prompts use Python-style pseudocode that maps to the concepts tested by all boards.

You need to understand the three basic programming constructs: sequence (executing instructions in order), selection (making decisions using if/else statements), and iteration (repeating actions using for loops and while loops). Beyond these fundamentals, you should be comfortable with using subroutines (functions and procedures) to create modular, reusable code, and understand the difference between local and global variables.

Practical programming skills are also tested, including string manipulation, file handling (reading from and writing to text files), input validation, and testing strategies such as using normal, boundary, erroneous, and invalid test data. Understanding the features of an integrated development environment (IDE), such as syntax highlighting, debugging tools, and auto-completion, is also part of the specification. These prompts cover all of these areas with interactive exercises.

Prompt 53: Variables, Constants, and Data Types

Copy this prompt into your AI tool:

Ask me to demonstrate my Computer Science knowledge. Explain the difference between a variable and a constant, with examples of when each should be used. Then explain the five main data types: integer, real (float), Boolean, character, and string. For each data type, give an example value and a scenario where it would be used. Quiz me with 8 quick questions: give me values and ask me to identify the data type, or describe a scenario and ask whether a variable or constant is appropriate. Mark each answer.

What this helps you practise:

Understanding variables, constants, and data types and when to use each.

How to use it well:

The difference between a variable and a constant is a common exam question. Remember: if the value should never change during the program, it should be a constant.

Prompt 54: Operators and Expressions

Copy this prompt into your AI tool:

*Set me a challenge: Explain the three types of operators used in programming: arithmetic operators (+, -, *, /, MOD, DIV, ^), comparison operators (==, !=, <, >, <=, >=), and Boolean operators (AND, OR, NOT). Give examples of each. Then give me 10 expressions and ask me to evaluate them, including expressions that combine multiple operator types. Present them one at a time, mark each answer, and explain the order of operations where relevant.*

What this helps you practise:

Evaluating arithmetic, comparison, and Boolean expressions correctly.

How to use it well:

Pay special attention to MOD (remainder) and DIV

(integer division) as these are commonly tested and frequently confused by students.

Prompt 55: Selection: If/Else Mastery

Copy this prompt into your AI tool:

Quiz me on GCSE Computer Science. Explain how selection works using IF, ELSE IF, and ELSE statements. Show me the pseudocode syntax and a worked example. Then give me 4 programming challenges that require selection: (1) a grade boundary checker, (2) a ticket price calculator based on age, (3) a simple menu system with 3 options, and (4) a leap year checker. Ask me to write pseudocode for each, then compare my solution with a model answer and point out any logic errors or missing edge cases.

What this helps you practise:

Writing selection statements (IF/ELSE IF/ELSE) to solve programming problems.

How to use it well:

Always consider edge cases and the order of your conditions. Test mentally with values at each boundary to check your logic.

Prompt 56: Iteration: For and While Loops

Copy this prompt into your AI tool:

Act as my GCSE Computer Science tutor. Explain the difference between count-controlled iteration (FOR loops) and condition-controlled iteration (WHILE loops). Show me the pseudocode syntax for each with examples. Then give me 5 programming tasks: 2 that require a FOR loop, 2 that require a WHILE loop, and 1 where I must choose the most appropriate type. Ask me to write the pseudocode, then check my solutions. Explain any errors and help me understand when to use each type.

What this helps you practise:

Writing FOR loops and WHILE loops and choosing the appropriate type for different situations.

How to use it well:

Use FOR when you know how many times the loop should repeat. Use WHILE when the loop should continue until a condition is met and you do not know in advance how many iterations this will take.

Prompt 57: Arrays and Lists

Copy this prompt into your AI tool:

Quiz me on GCSE Computer Science. Explain what an array (or list) is, how elements are accessed using indices (starting from 0), and common operations: adding, removing, accessing, and updating elements. Write pseudocode examples for each operation. Then give me 4 coding challenges using arrays: (1) find the sum of all elements, (2) find the maximum value, (3) count how many elements meet a condition, and (4) reverse the array. Ask me to write pseudocode for each, then mark and correct my solutions.

What this helps you practise:

Using arrays/lists to store and manipulate collections of data.

How to use it well:

Remember that array indices typically start at 0. Off-by-one errors are one of the most common programming mistakes.

Prompt 58: Subroutines: Functions and Procedures

Copy this prompt into your AI tool:

Act as my GCSE Computer Science tutor. Explain the difference between a function and a procedure. Cover how to define and call subroutines, how to pass parameters, and the difference between local

and global variables. Give me pseudocode examples of each. Then give me 3 problems and ask me to write subroutines to solve them: (1) a function that takes two numbers and returns the larger one, (2) a procedure that prints a greeting using a name parameter, and (3) a function that checks if a number is prime. Mark each solution and explain any errors.

What this helps you practise:

Writing and using functions and procedures with parameters and return values.

How to use it well:

Remember: functions return a value, procedures do not. Using subroutines makes code modular, reusable, and easier to test, which are points examiners want to see.

Prompt 59: String Manipulation Techniques

Copy this prompt into your AI tool:

Test me on this Computer Science topic. Teach me common string manipulation operations: finding the length, extracting substrings, converting case, concatenation, and converting between strings and numbers. Show me pseudocode or Python examples of each. Then give me 5 string manipulation challenges: (1) extract the first 3 characters of a string, (2) check if a string is a palindrome, (3) count the vowels in a string, (4) reverse a string, and (5) replace all spaces with hyphens. Ask me to write code for each and mark my solutions.

What this helps you practise:

Performing common string manipulation operations in code.

How to use it well:

String manipulation questions appear frequently in programming exams. Practise the most common operations until they feel automatic.

Prompt 60: File Handling: Reading and Writing

Copy this prompt into your AI tool:

Act as my GCSE Computer Science tutor. Explain how to read from and write to text files in pseudocode, covering opening a file, reading line by line, writing data, appending data, and closing a file. Show me the pseudocode syntax for each operation. Then give me 3 programming tasks involving files: (1) read a file and display its contents, (2) write user input to a new file, and (3) read a file of numbers and calculate the average. Ask me to write the pseudocode, then mark my answers and correct any file handling errors.

What this helps you practise:

Reading from and writing to text files using correct file handling techniques.

How to use it well:

Always remember to close files after use. In the exam, show that you understand the difference between reading, writing, and appending modes.

Prompt 61: Input Validation Techniques

Copy this prompt into your AI tool:

Quiz me on GCSE Computer Science. Explain what input validation is and why it is important. Describe these validation checks: range check, type check, length check, presence check, and format check. Give a real-world example of each. Then give me 4 scenarios and ask me to write pseudocode that validates user input using appropriate checks, including a while loop that keeps asking until valid input is provided. Mark each solution and suggest improvements.

What this helps you practise:

Implementing input validation using appropriate checks and loops.

How to use it well:

Validation is about checking that input meets certain criteria before processing it. Always use a WHILE loop to repeatedly ask for input until valid data is entered.

Prompt 62: Testing with Normal, Boundary, and Erroneous Data

Copy this prompt into your AI tool:

Set me a challenge: Explain the difference between normal test data, boundary test data, and erroneous test data. For a program that accepts ages between 11 and 18, ask me to suggest examples of each type of test data and explain what the expected outcome should be. Then give me 3 more programs with different validation rules and ask me to create a test plan for each, listing the test data, the type of test, and the expected result. Mark my test plans and identify any gaps.

What this helps you practise:

Creating comprehensive test plans using normal, boundary, and erroneous test data.

How to use it well:

Boundary testing is the area most students forget. Always test the exact boundary values and one either side of them.

Prompt 63: IDE Features and Debugging

Copy this prompt into your AI tool:

Act as my GCSE Computer Science tutor. Explain the key features of an integrated development environment (IDE) that help programmers write and debug code: syntax highlighting, auto-completion, error diagnostics, breakpoints, stepping through code, variable watch windows, and code editors. For each feature, explain how it helps the programmer. Then quiz me with 5 questions where you describe a

problem a programmer is having and I identify which IDE feature would help most. Mark each answer.

What this helps you practise:

Understanding IDE features and how they support programming and debugging.

How to use it well:

Exam questions often describe a scenario and ask which IDE feature would help. Focus on understanding the purpose of each feature rather than just its name.

Prompt 64: Common Programming Errors

Copy this prompt into your AI tool:

You are a GCSE Computer Science examiner. Explain the difference between syntax errors, logic errors, and runtime errors. Give me two examples of each in pseudocode. Then show me 6 short code snippets, each containing one error, and ask me to identify the type of error, explain what is wrong, and fix the code. Present them one at a time, mark each answer, and provide the corrected code with an explanation. Emphasise which errors the IDE can detect and which require testing.

What this helps you practise:

Identifying and fixing syntax errors, logic errors, and runtime errors in code.

How to use it well:

Syntax errors are caught by the IDE, but logic errors only show up through testing. Develop the habit of tracing through your code mentally to find logic errors.

Prompt 65: Nested Iteration and Selection

Copy this prompt into your AI tool:

Present me with a Computer Science challenge. Explain what nested iteration (a loop inside a loop)

and nested selection (an if statement inside another if statement) mean. Give me a worked example of each in pseudocode. Then challenge me with 3 problems that require nested structures: (1) print a multiplication table, (2) search a 2D array for a value, and (3) a menu system that loops until the user chooses to exit, with input validation on each choice. Ask me to write pseudocode, then mark and correct my solutions.

What this helps you practise:

Writing programs that use nested loops and nested selection statements.

How to use it well:

Indentation is critical when nesting structures. Make sure each level of nesting is clearly indented so you can track which code belongs to which loop or condition.

Prompt 66: Programming Concepts Exam Practice

Copy this prompt into your AI tool:

Set me a challenge: Create a programming mini exam with 6 questions: (1) write a function that returns the factorial of a number, (2) trace through a given pseudocode algorithm using a trace table, (3) identify and fix an error in a code snippet, (4) write pseudocode with input validation, (5) explain the difference between local and global variables, and (6) write a program that reads data from a file and processes it. Present each one at a time, mark my answers, and give model solutions. Provide a final score.

What this helps you practise:

Applying a range of programming concepts under exam conditions.

How to use it well:

Time yourself: aim for about 3 minutes per short

question and 6 minutes for the longer coding questions. This builds exam pace.

Section 6

Boolean Logic and Data Structures

Boolean logic forms the mathematical foundation of how computers make decisions. At GCSE level, you need to understand the three basic logic gates (AND, OR, NOT) and be able to construct and interpret truth tables for expressions involving these gates. You should also be able to combine multiple gates into logic circuits and determine the output for any given combination of inputs.

Being able to write and simplify Boolean expressions is another important skill. You need to recognise how Boolean expressions map to logic circuits and vice versa, and how these relate to the selection statements you use in programming. Understanding that every digital decision a computer makes is ultimately based on Boolean logic helps connect this theoretical topic to practical computing.

Data structures are the ways in which data is organised and stored so it can be used efficiently. At GCSE, you need to understand records as a way of storing related data fields, and you must be familiar with the basics of SQL (Structured Query Language) for querying databases. Being able to write SELECT statements with WHERE clauses to retrieve specific data from a table is a commonly examined skill.

Prompt 67: AND, OR, NOT Gates Explained

Copy this prompt into your AI tool:

Quiz me on GCSE Computer Science. Explain how each of the three basic logic gates works: AND, OR, and NOT. For each gate, describe its function in plain English, show its truth table, and give a real-world analogy (such as light switches or security systems). Then test me with 8 quick questions where

I determine the output of a single gate for given inputs. Present each one at a time and mark my answer immediately. Track my score and identify which gate I need more practice with.

What this helps you practise:

Understanding AND, OR, and NOT logic gates and their truth tables.

How to use it well:

Memorise the truth tables for all three gates. AND outputs 1 only when all inputs are 1. OR outputs 1 when any input is 1. NOT simply inverts the input.

Prompt 68: Truth Table Construction

Copy this prompt into your AI tool:

You are a GCSE Computer Science examiner. Teach me how to construct truth tables for Boolean expressions involving two or three inputs. Start with simple expressions like A AND B, then progress to more complex ones like (A OR B) AND NOT C. Show me how to set up the columns, list all possible input combinations, and evaluate the expression step by step. Then give me 3 Boolean expressions and ask me to construct the truth table for each. Check my tables and correct any errors.

What this helps you practise:

Constructing truth tables for Boolean expressions with multiple inputs and operators.

How to use it well:

For n inputs, there are 2^n rows in the truth table. Use intermediate columns for sub-expressions to avoid mistakes. Work through each row systematically.

Prompt 69: Boolean Expression Writing

Copy this prompt into your AI tool:

Present me with a Computer Science challenge. Show me how to write Boolean expressions from

word descriptions and from logic circuit diagrams (described in text). Give me 3 worked examples. Then test me: give me 4 word descriptions such as 'the output is true when A is true and either B or C is true' and ask me to write the Boolean expression. Also give me 2 logic circuit descriptions and ask me to write the matching expression. Mark each answer and show the correct expression if I am wrong.

What this helps you practise:

Writing Boolean expressions from descriptions and logic circuits.

How to use it well:

Translate the words directly: 'and' becomes AND, 'or' becomes OR, 'not' becomes NOT. Use brackets to clarify the order of operations.

Prompt 70: Logic Circuit Challenge

Copy this prompt into your AI tool:

You are a GCSE Computer Science examiner. Describe 3 logic circuits in text form (for example: 'Input A and Input B go into an AND gate, the output of this AND gate and Input C go into an OR gate'). For each circuit, ask me to: (1) write the Boolean expression, (2) construct the truth table, and (3) determine the output for a specific set of inputs. Check each part of my answer and explain any errors. Then give me a Boolean expression and ask me to describe the corresponding logic circuit.

What this helps you practise:

Interpreting logic circuits, writing Boolean expressions, and completing truth tables.

How to use it well:

Work through logic circuits from the inputs towards the output, evaluating each gate in turn. This systematic approach prevents errors.

Prompt 71: Boolean Logic in Programming

Copy this prompt into your AI tool:

Give me a GCSE-style question about Computer Science. Explain how Boolean logic connects to programming selection statements. Show me how an IF statement like 'IF age >= 18 AND hasTicket == true' is a Boolean expression that evaluates to true or false. Give me 4 real-world conditions and ask me to write them as Boolean expressions and then as IF statements in pseudocode. Then give me 3 IF statements and ask me to identify the Boolean expression within each and evaluate it for given variable values. Mark each answer.

What this helps you practise:

Connecting Boolean logic to programming selection statements.

How to use it well:

Every IF statement contains a Boolean expression. Practise spotting and evaluating these expressions, especially when they use AND, OR, and NOT together.

Prompt 72: Records and Data Structures

Copy this prompt into your AI tool:

Set me a challenge: Explain what a record is in the context of data structures, including how records store multiple fields of different data types for a single entity. Give me an example of a student record with fields like name, age, and grade. Show me how to define and access records in pseudocode. Then give me 3 scenarios and ask me to design an appropriate record structure for each, choosing suitable field names and data types. Mark my designs and suggest improvements.

What this helps you practise:

Designing record data structures with appropriate fields and data types.

How to use it well:

Think of a record as a row in a table, where each column is a field. Choose data types carefully: names are strings, ages are integers, prices are real numbers.

Prompt 73: SQL Basics: SELECT, FROM, WHERE

Copy this prompt into your AI tool:

Quiz me on GCSE Computer Science. Teach me the basics of SQL for GCSE. Explain the syntax of a SELECT statement including SELECT, FROM, WHERE, and ORDER BY clauses. Show me a sample database table (described in text) with at least 6 records. Then give me 6 SQL challenges of increasing difficulty: from selecting all records, to selecting specific fields, using WHERE with conditions, using AND/OR in WHERE clauses, and ordering results. Ask me to write the SQL query for each. Mark each query and correct any syntax errors.

What this helps you practise:

Writing SQL SELECT queries with WHERE clauses to retrieve data from databases.

How to use it well:

SQL keywords are conventionally written in uppercase (SELECT, FROM, WHERE) although they are not case-sensitive. Practice the exact syntax as it will be marked precisely in exams.

Prompt 74: Boolean Logic and SQL Combined Quiz

Copy this prompt into your AI tool:

Present me with a Computer Science challenge. Give me a combined quiz covering Boolean logic and SQL with 8 questions: 2 truth table questions, 2 Boolean expression questions, 2 logic circuit questions, and 2

SQL query questions. Present each one at a time, mark my answer, and provide the model answer. At the end, give me a score out of 8 and identify which specific area within this section I should focus on for further revision.

What this helps you practise:

Testing knowledge across Boolean logic and SQL in a mixed quiz format.

How to use it well:

This quiz simulates how these topics might appear together in an exam. Practise switching between different types of questions to build mental flexibility.

Section 7

Fixing Common Mistakes and Misconceptions

Every GCSE Computer Science student makes mistakes during revision, and some misconceptions are so common that examiners see them in hundreds of scripts every year. This section is designed to target the most frequent errors and help you identify, understand, and correct them before the exam. Addressing these mistakes now can be worth several marks on exam day.

Common mistakes include confusing RAM with ROM, misunderstanding the difference between lossy and lossless compression, making errors in binary conversions, and struggling with the efficiency differences between algorithms. These are not signs of weakness; they are predictable stumbling blocks that affect students at all ability levels. The key is deliberate practice that targets these specific areas.

The prompts in this section take a different approach from the rest of the book. Instead of teaching you new content, they present you with incorrect statements, flawed code, or common exam answer mistakes and ask you to spot and fix the errors. This active error-correction approach is one of the most effective ways to strengthen your understanding and avoid losing marks to careless mistakes in the real exam.

Prompt 75: RAM vs ROM Misconception Buster

Copy this prompt into your AI tool:

Quiz me on GCSE Computer Science. Present me with 6 statements about RAM and ROM, some correct and some containing common misconceptions (such as 'ROM is used to store user

files' or 'RAM stores data permanently'). Ask me to identify whether each statement is true or false, and to correct any false statements. After each answer, confirm the correct response and explain the misconception. Finish by asking me to write a clear two-sentence explanation of each that I could use in an exam.

What this helps you practise:

Identifying and correcting common misconceptions about RAM and ROM.

How to use it well:

The most common error is confusing volatility with read/write capability. RAM is volatile (loses data when power is off) and read/write. ROM is non-volatile and read-only.

Prompt 76: Binary Conversion Error Spotter

Copy this prompt into your AI tool:

Present me with a Computer Science challenge. Show me 6 binary-to-denary and denary-to-binary conversions that a student has attempted, some correct and some containing common mistakes (such as using the wrong place values, forgetting to include leading zeros, or making arithmetic errors). Ask me to check each conversion, identify whether it is correct, and fix any errors by showing the correct working. After each, explain the mistake the student made so I can avoid it.

What this helps you practise:

Spotting and correcting common errors in binary and denary conversions.

How to use it well:

Always double-check your conversions by converting back the other way. If you convert denary to binary, verify by converting the binary result back to denary.

Prompt 77: Lossy vs Lossless Confusion Fix

Copy this prompt into your AI tool:

Ask me to demonstrate my Computer Science knowledge. Present me with 5 statements about compression that contain common student errors or ambiguous wording (such as 'lossy compression removes data so it is always bad' or 'lossless compression keeps all the quality so files stay the same size'). Ask me to identify the error in each statement and rewrite it correctly. After each, explain why students commonly make this mistake and what the precise correct understanding is.

What this helps you practise:

Correcting misconceptions about lossy and lossless compression.

How to use it well:

The key distinction is that lossy permanently removes data to achieve smaller files, while lossless allows perfect reconstruction of the original. Neither is universally better.

Prompt 78: Algorithm Efficiency Myth Busting

Copy this prompt into your AI tool:

Act as my GCSE Computer Science tutor. Present me with 5 common student misconceptions about algorithm efficiency (such as 'binary search is always faster than linear search' or 'bubble sort and merge sort take the same time for small lists'). Ask me to explain why each statement is misleading or incorrect. After each answer, give me the correct nuanced explanation. Then ask me a 4-mark exam question about comparing two algorithms where I need to show balanced understanding.

What this helps you practise:

Correcting misconceptions about algorithm efficiency and making accurate comparisons.

How to use it well:

Remember that binary search requires sorted data,

which has its own cost. And for very small datasets, simpler algorithms like bubble sort may actually be faster due to lower overhead.

Prompt 79: Programming Logic Error Detective

Copy this prompt into your AI tool:

You are a GCSE Computer Science examiner. Show me 5 short pseudocode programs, each containing a logic error (not a syntax error). The errors should include: an infinite loop, an off-by-one error, a wrong comparison operator, an uninitialised variable, and an incorrect loop boundary. For each program, tell me what the program is supposed to do, then ask me to find the logic error, explain what goes wrong, and write the corrected code. Mark each answer and show the correction if I miss anything.

What this helps you practise:

Finding and fixing logic errors in pseudocode programs.

How to use it well:

Use a trace table to work through each program step by step. Logic errors are invisible to the IDE and can only be found through careful analysis or testing.

Prompt 80: Network Terminology Mix-Ups

Copy this prompt into your AI tool:

Act as my GCSE Computer Science tutor. Present me with 6 statements about networks that contain commonly confused terminology (such as mixing up routers and switches, confusing protocols with topologies, or misusing the terms packet and frame). Ask me to spot the incorrect term in each statement and replace it with the correct one. After each, explain the difference between the confused terms. Finish with a quick matching exercise: 8 terms matched to 8 definitions.

What this helps you practise:
Correcting commonly confused networking terminology.
How to use it well:
Create a glossary of network terms with precise definitions. Many exam marks are lost simply because students use the wrong technical term.

Prompt 81: Data Type and Variable Mistakes
Copy this prompt into your AI tool:
Give me a GCSE-style question about Computer Science. Show me 5 code snippets where a student has used the wrong data type, confused a variable with a constant, or made a casting error (such as trying to add a string to an integer without conversion). For each snippet, ask me to identify the mistake, explain what would go wrong when the program runs, and write the corrected version. Mark each answer. Then ask me to explain why choosing the correct data type matters for validation and storage.
What this helps you practise:
Identifying and correcting data type errors and variable misuse in code.
How to use it well:
Always declare variables with the correct data type and convert types explicitly when needed. Type mismatch errors are common in exam code-tracing questions.

Prompt 82: Exam Answer Upgrade Workshop
Copy this prompt into your AI tool:
Act as my GCSE Computer Science tutor. Show me 4 weak exam answers that a student might write for common GCSE Computer Science questions (covering topics like explaining how the internet transmits data, comparing storage types, describing

the purpose of an operating system, and explaining why validation is needed). For each weak answer, ask me to identify what marks it would lose and how to improve it. Then show me the model answer and explain what makes it worth full marks.

What this helps you practise:

Identifying weaknesses in exam answers and improving them to gain full marks.

How to use it well:

Compare the weak and strong answers carefully. Notice how the strong answers use precise terminology, give specific examples, and address every part of the question.

Prompt 83: Hexadecimal and Binary Pitfalls

Copy this prompt into your AI tool:

Test me on this Computer Science topic. Present me with 5 hexadecimal conversion attempts that contain common student errors, such as: forgetting that hex digits go from 0 to F (not 0 to 9), miscalculating nibble values, or confusing the direction of conversion. Ask me to check each one, find the error, and produce the correct answer with working. After all 5, give me a summary of the most important things to remember when working with hexadecimal.

What this helps you practise:

Spotting and correcting common hexadecimal conversion errors.

How to use it well:

Remember: A=10, B=11, C=12, D=13, E=14, F=15. Always split binary into nibbles (groups of 4) from the right when converting to hex.

Prompt 84: Misconceptions Speed Round

Copy this prompt into your AI tool:

Quiz me on GCSE Computer Science. Give me a speed round of 10 true-or-false statements covering common misconceptions across the entire GCSE Computer Science specification. Include misconceptions about hardware, software, networks, algorithms, programming, and data representation. Give me each statement one at a time and ask for true or false with a brief justification. Mark each answer immediately. At the end, give me a score and explain any I got wrong with the correct understanding.

What this helps you practise:

Rapidly identifying common misconceptions across all GCSE Computer Science topics.

How to use it well:

This is a great warm-up exercise before a revision session. It quickly identifies areas where misconceptions might be lurking in your understanding.

Section 8

Exam Technique and Extended Responses

Knowing the content is only half the battle in GCSE Computer Science. You also need strong exam technique to translate your knowledge into marks. Understanding command words (define, describe, explain, compare, evaluate, discuss) and knowing what each one requires in your answer can make the difference between a good grade and a great one.

Extended response questions, worth 6 or more marks, require you to structure your answer clearly, use accurate technical terminology, and provide detailed explanations with relevant examples. Many students lose marks not because they lack knowledge but because they do not answer the question that was asked, or they do not write enough detail to earn all available marks.

This section focuses on building your exam technique through practice with different question types. You will practise writing under timed conditions, structuring extended responses, decoding what examiners are looking for, and avoiding the common pitfalls that cost marks. These skills are just as important as subject knowledge and deserve dedicated revision time.

Prompt 85: Command Words Decoded

Copy this prompt into your AI tool:

Test me on this Computer Science topic. Explain what each of these exam command words requires in an answer: State, Define, Describe, Explain, Compare, Evaluate, Discuss, and Justify. For each, tell me how much detail is expected and give me an example of how to structure a response. Then give me 6 questions using different command words on

the topic of computer networks. After I answer each one, mark it specifically based on whether I responded appropriately for the command word used, not just whether the content was correct.

What this helps you practise:

Understanding exam command words and structuring answers appropriately for each.

How to use it well:

Highlight the command word in every exam question before you start writing. This tells you exactly what type of answer the examiner expects.

Prompt 86: Writing Pseudocode Under Pressure

Copy this prompt into your AI tool:

Act as my GCSE Computer Science tutor. Give me 4 pseudocode writing tasks of increasing difficulty, each with a strict time guide: Task 1 (3 minutes): a simple input-process-output program. Task 2 (4 minutes): a program using selection and iteration. Task 3 (5 minutes): a program with subroutines and parameters. Task 4 (6 minutes): a complete program with file handling and validation. After each, mark my pseudocode for correctness, clarity, and use of appropriate constructs. Identify any marks I would lose and show the model answer.

What this helps you practise:

Writing correct pseudocode efficiently under timed exam conditions.

How to use it well:

Practise planning your pseudocode for 30 seconds before writing. A quick plan of inputs, processes, and outputs prevents you from getting stuck halfway through.

Prompt 87: Structured Explanation Builder

Copy this prompt into your AI tool:

Set me a challenge: Give me 4 'Explain' questions worth 4 marks each, covering different topics: (1) explain how data is transmitted across a network using packet switching, (2) explain why solid-state storage is suitable for a laptop, (3) explain how the CPU uses the fetch-decode-execute cycle, and (4) explain why input validation is important in programming. For each, ask me to write my answer, then mark it against the mark scheme criteria. Show me how to structure each answer using point-explanation-example format.

What this helps you practise:

Writing structured explanations that earn full marks in 4-mark questions.

How to use it well:

For a 4-mark explain question, aim for 4 distinct points, each with a clear explanation. Use connective words like 'because', 'this means that', and 'as a result' to show chains of reasoning.

Prompt 88: Compare and Contrast Technique

Copy this prompt into your AI tool:

Act as my GCSE Computer Science tutor. Teach me how to structure a comparison answer effectively. Then give me 4 'Compare' questions: (1) compare RAM and ROM, (2) compare linear search and binary search, (3) compare lossy and lossless compression, and (4) compare star and mesh topologies. For each, ask me to write a structured comparison. Mark each answer, checking that I have addressed both items in each point rather than writing about each separately. Show me the model answer format.

What this helps you practise:

Writing effective comparison answers that directly contrast two items.

How to use it well:

Use linking phrases like 'whereas', 'in contrast', 'on the other hand', and 'however' to make direct comparisons. Avoid writing two separate descriptions.

Prompt 89: 6-Mark Extended Response Practice

Copy this prompt into your AI tool:

Set me a challenge: Give me 3 extended response questions worth 6 marks each: (1) Discuss how an organisation can protect its network from cyber security threats, (2) Evaluate the use of cloud storage versus local storage for a school, (3) Explain how the choice of data representation affects the quality and file size of digital media. For each, let me write my full answer. Then mark it using the levels-based mark scheme approach: Level 1 (1-2 marks) for basic points, Level 2 (3-4 marks) for detailed explanation, Level 3 (5-6 marks) for comprehensive and well-structured responses with technical terminology. Show the model answer.

What this helps you practise:

Writing high-quality extended responses that achieve top-level marks.

How to use it well:

Plan your answer before writing: list 3-4 key points, then develop each with an explanation and example. Use a clear paragraph structure and precise terminology throughout.

Prompt 90: Evaluate Questions Strategy

Copy this prompt into your AI tool:

Give me a GCSE-style question about Computer Science. Explain what an 'Evaluate' question requires: weighing up advantages and disadvantages and reaching a justified conclusion. Then give me 3 evaluate questions: (1) Evaluate the use of biometric

authentication versus password-based authentication, (2) Evaluate whether open-source software or proprietary software is better for a small business, and (3) Evaluate the impact of increasing the sample rate when recording digital audio. For each, mark my answer on whether I have given balanced arguments and reached a justified conclusion.

What this helps you practise:

Writing balanced evaluate answers with justified conclusions.

How to use it well:

Structure your answer as: advantages of option A, disadvantages of option A, advantages of option B, disadvantages of option B, conclusion with justification. The conclusion must be supported by your arguments.

Prompt 91: Mark Scheme Reverse Engineering

Copy this prompt into your AI tool:

Act as my GCSE Computer Science tutor. Show me 3 exam questions and their corresponding model answers with mark allocations. For each, ask me to act as the examiner: I should identify what each mark is awarded for and explain why certain phrases or terms earn marks. After I analyse each mark scheme, give me a similar question and ask me to write an answer that hits every mark point. Then mark my answer against the criteria I just identified.

What this helps you practise:

Understanding how mark schemes work so you can target specific marks in your answers.

How to use it well:

Studying mark schemes is one of the best revision strategies. Notice that marks are awarded for specific technical terms, clear explanations, and relevant examples.

Prompt 92: Time Management in the Exam

Copy this prompt into your AI tool:

Ask me to demonstrate my Computer Science knowledge. Help me create a time management plan for my Computer Science exam. Ask me how long the exam is and how many marks it is worth, then calculate how many minutes per mark I have. Give me strategies for: how long to spend reading the question, when to move on from a difficult question, how to use spare time at the end, and how to allocate time for extended response questions. Then give me a mock mini-paper with 5 questions of different mark values and ask me to plan my time before attempting them.

What this helps you practise:

Planning and managing time effectively during the Computer Science exam.

How to use it well:

A common rule is roughly one minute per mark plus reading time. For a 1.5-hour exam worth 80 marks, that is just over 1 minute per mark. Plan before you start writing.

Prompt 93: Avoiding Common Exam Pitfalls

Copy this prompt into your AI tool:

You are a GCSE Computer Science examiner. Describe the 8 most common mistakes students make in Computer Science exams: (1) not reading the question carefully, (2) writing too little for the marks available, (3) using vague language instead of technical terms, (4) repeating the question in the answer, (5) confusing describe and explain, (6) not showing working in calculations, (7) leaving questions blank, and (8) poor time management. For each pitfall, give me a specific example and a

strategy to avoid it. Then test me with a question where I am likely to fall into one of these traps.

What this helps you practise:

Recognising and avoiding common exam mistakes that cost marks.

How to use it well:

Read this prompt early in your revision and revisit it the week before the exam. Being aware of these pitfalls makes you much less likely to fall into them.

Prompt 94: Exam Technique Speed Drill

Copy this prompt into your AI tool:

Act as my GCSE Computer Science tutor. Give me 8 exam questions in quick succession, mixing short-answer (1-2 marks) and medium-answer (3-4 marks) questions from across the specification. For each question, I must: (1) identify the command word, (2) decide how many points I need to make, and (3) write a concise answer. Mark each answer immediately, focusing on whether my response matches the demands of the command word and mark allocation. Give me a final score and identify any technique issues.

What this helps you practise:

Practising rapid question analysis and appropriately scaled answers across the specification.

How to use it well:

This drill builds the skill of quickly analysing what a question requires. In the real exam, spending 10 seconds understanding the question saves minutes of writing the wrong type of answer.

Section 9

Final Revision and Exam-Week Prompts

The final days before your GCSE Computer Science exam are not the time to learn new material. Instead, this is when you consolidate what you already know, fill any last remaining gaps, and build your confidence through targeted practice. These prompts are designed specifically for exam week, when you need efficient, focused revision that covers the maximum ground in minimum time.

The prompts in this section include a rapid whole-specification sweep that touches every major topic in one session, focused drills on the calculations that come up most often (binary conversions, file size calculations, and truth tables), and a mini mock that simulates exam conditions. These are designed to be used in the three to five days before your exam.

Your goal in exam week is not perfection but readiness. Use these prompts to confirm what you know, identify any final weak spots, and practise retrieving information under mild pressure. Stay calm, trust your preparation, and remember that consistent revision over weeks and months matters far more than last-minute cramming. These final prompts will help you walk into the exam room feeling prepared and confident.

Prompt 95: Whole-Specification Speed Sweep

Copy this prompt into your AI tool:

Present me with a Computer Science challenge. Take me through a rapid review of the entire GCSE Computer Science specification. For each of the following topics, ask me one key question that tests the most important concept: data representation, computer architecture, networks, cyber security,

algorithms, programming, Boolean logic, and databases. Give me each question one at a time. After each answer, give me a quick verdict (strong, acceptable, or needs work) and a one-sentence correction if needed. At the end, list the topics I should prioritise in my remaining revision time.

What this helps you practise:

Rapidly reviewing all major topics to identify strengths and weaknesses before the exam.

How to use it well:

Use this 2-3 days before the exam to identify your final revision priorities. Spend your remaining time on the topics marked as 'needs work'.

Prompt 96: Conversion Drill: Binary, Denary, Hexadecimal

Copy this prompt into your AI tool:

Test me on this Computer Science topic. Give me a rapid-fire drill of 12 number conversion questions, alternating between: binary to denary, denary to binary, binary to hexadecimal, hexadecimal to denary, denary to hexadecimal, and hexadecimal to binary. Present each one at a time, give me a moment to answer, then mark it immediately. Move quickly. At the end, give me my score out of 12, my average speed, and tell me which conversion direction I should practise once more before the exam.

What this helps you practise:

Building speed and accuracy in number system conversions under time pressure.

How to use it well:

Do this drill daily in the final week. Speed matters because quick conversions free up time for harder questions in the exam.

Prompt 97: File Size and Storage Calculations

Copy this prompt into your AI tool:

Present me with a Computer Science challenge. Give me 6 calculation questions covering all the main file size formulas: (1) image file size from width, height, and colour depth, (2) sound file size from sample rate, bit depth, and duration, (3) text file size from number of characters and encoding (ASCII or Unicode), (4) converting between units (bits to bytes to KB to MB), (5) how many files of a given size fit on a storage device, and (6) the effect of compression on file size. Present each one at a time, mark my answer, and show the full working. Give me a score at the end.

What this helps you practise:

Practising all file size calculations that appear in the exam.

How to use it well:

Write down all the formulas on a revision card and review them before attempting this drill. Show all your working as marks are awarded for method even if the final answer is wrong.

Prompt 98: Mini Mock Exam

Copy this prompt into your AI tool:

Give me a GCSE-style question about Computer Science. Give me a 30-minute mini mock exam with the following structure: 4 short-answer questions worth 2 marks each (covering data representation, systems architecture, networks, and programming), 2 medium-answer questions worth 4 marks each (one on algorithms and one on Boolean logic), and 1 extended response question worth 6 marks (on cyber security). Present all the questions at once so I can manage my time. After I submit all my answers, mark each one against a detailed mark scheme and give me a total score out of 22. Provide feedback on both my subject knowledge and exam technique.

What this helps you practise:

Completing a timed mini mock exam covering the full specification.

How to use it well:

Set a timer for 30 minutes and attempt all questions without looking at notes. This simulates exam pressure and helps you practise time management.

Prompt 99: Key Definitions Rapid Recall

Copy this prompt into your AI tool:

You are a GCSE Computer Science examiner. Test me on 15 key definitions that frequently appear in GCSE Computer Science exams. Include: algorithm, decomposition, abstraction, variable, constant, validation, authentication, encryption, malware, phishing, protocol, packet switching, fetch-decode-execute cycle, Boolean, and subroutine. Give me each term one at a time and ask me to define it in one or two sentences. Mark each definition as complete, partial, or incorrect, and provide the model definition. Give me a final score and list any definitions I need to memorise before the exam.

What this helps you practise:

Rapidly recalling precise definitions of key Computer Science terminology.

How to use it well:

Use this the night before the exam or on the morning of the exam as a final knowledge check. Precise definitions earn marks in short-answer questions.

Prompt 100: Final Confidence Check and Exam Readiness

Copy this prompt into your AI tool:

Ask me to demonstrate my Computer Science knowledge. Ask me to rate my confidence from 1 to 5 on each of these topics: data representation, computer systems, networks, cyber security,

algorithms, programming, Boolean logic, and databases. Based on my ratings, give me a personalised 30-minute revision plan targeting my lowest-confidence topics. Then give me one challenging question from each of my two weakest topics to test whether my low confidence is justified or whether I know more than I think. Mark my answers and give me an honest assessment of my readiness, along with three pieces of advice for exam day.

What this helps you practise:

Assessing overall exam readiness and creating a final targeted revision plan.

How to use it well:

Use this as your very last revision activity. Be honest with your confidence ratings so the AI can give you the most useful final advice.

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 GCSE.

Remember: the goal was never to rely on AI for answers. The goal was to use it 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 mistake you identified, each explanation you improved, and each gap you filled has strengthened your thinking.

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

- Add constraints (for example, “limit to three key points”).
- Increase difficulty gradually.
- Ask the AI to challenge your reasoning.
- Request alternative explanations.
- Ask it to critique your thinking rather than provide answers.

The most powerful use of AI is not asking it to tell you things — it is asking it to test and refine your thinking.

In the future, those who understand how to use tools intelligently will have an advantage. Treat AI as a tutor, not a shortcut. The skill of asking better questions will continue to matter long after your exams are over.

About the Author

James R. Martin holds an MSci in Physics from the University of Bristol and a PGCE with a Physics focus from the University of Oxford. He has over a decade of experience teaching and tutoring students aged 11–18 across a range of subjects, including Physics, Biology, Chemistry, Mathematics, Economics, and Electronics.

He has worked with multiple syllabi, including GCSE, A-Level, KS3, and the International Baccalaureate Diploma Programme (IBDP), supporting students of varying abilities to develop clarity, confidence, and exam success.

His work focuses on effective revision strategies, independent thinking, and the responsible use of artificial intelligence as a tool to strengthen — not replace — understanding.

Other Titles in This Series

The *100 AI Prompts for Smarter Revision* series supports students across GCSE, A-Level, and IBDP subjects.

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