

How to get a grade 9 in OCR GCSE Computer Science

A grade 9 means dropping very few marks anywhere. You need exact technical language that matches what mark schemes reward, Section B code that is correct, complete and robust, and extended response answers that are balanced, applied to the scenario and reach a clear judgement. At this level careless slips cost you as much as gaps in knowledge.

What a grade 9 asks of you

- Know the whole specification in depth, including the points students skim: embedded systems, character sets, metadata, the concept of layers, utility software, open source and proprietary licences, and the tools in an IDE.
- Write explanations as a chain of cause, effect and consequence, e.g. 'A larger cache holds more frequently used instructions, so fewer have to be fetched from slower RAM, so the CPU waits less and processes instructions faster.'
- Write Section B code that works first time on paper: correct logic, validated input, sensible identifiers, correct use of arrays, files and sub programs, and no off-by-one errors.
- Write extended response answers at the top level every time: technical, applied, balanced and concluded.
- Make no careless slips in calculations, trace tables or truth tables.

The detail that separates an 8 from a 9

- CPU performance: extra cores only help when the software can split its work between them, and some tasks must be done in sequence; a higher clock speed also produces more heat.
- Virtual memory: it lets more programs be open than RAM alone could hold, but moving data between RAM and secondary storage is slow, so performance drops.
- Compression: justify the type by the file. Lossless for text and program code, where every bit must be restored exactly; lossy for images, sound and video, where a much smaller file matters more than detail you are unlikely to notice.
- Networks: why a MAC address is needed as well as an IP address; how layers let each part of a network system be developed and changed separately; why standards let hardware and software from different makers work together.
- Algorithms: compare efficiency in terms of comparisons or passes, and state the conditions (binary search needs sorted data; merge sort needs extra memory).
- Binary: a left shift can cause overflow and a right shift loses the bits that fall off the end, so precision is lost. Be ready to explain both.
- Testing: choose test data that proves each boundary works, and explain how each validation check prevents a specific kind of bad input.

Question types to master

- The longest Section B questions: a full program from a written description, combining validation, loops, arrays or files, and a sub program.
- Refining code to make it more robust, efficient or reusable while keeping it working.
- Trace tables for nested loops, string slicing and arrays, where one slip spoils every row after it.
- Extended response discuss and evaluate questions in unfamiliar scenarios.
- Explain questions worth 4 to 6 marks that need several linked steps.
- Comparing algorithms, storage devices or network designs for a given situation, with a justified recommendation.

Revision plan, week by week

- This plan assumes about 10 weeks. With less time, start with the audit in week 1, then spend what remains on your weakest areas and the mocks.
- Week 1: sit a timed mock of each paper and go through the grade-by-grade checklist up to grade 9. This is your audit: it tells you exactly what to fix.
- Weeks 2 to 6: each week, one Paper 1 topic and one Paper 2 topic in depth, a Hard or Extreme custom paper on each, and one Section B style program written by hand.
- Weeks 2 to 8: one timed extended response answer a week on a new scenario.
- Weeks 7 and 8: mixed Extreme papers across all topics, and redo every question you dropped marks on.
- Week 9: a second timed mock of each paper. Aim to finish each with 10 minutes to spare.
- Week 10: polish with cheatsheets, flashcards and your personal list of slips, plus one short timed paper a day.

Top-level extended response answers

- Plan for 2 minutes: choose your angles (legal, ethical, cultural, environmental, privacy, or benefits against drawbacks) and the people in the scenario who are affected.
- Write four or five developed paragraphs, each with a point, an explanation and a consequence for someone in the scenario.
- Bring in accurate technical knowledge where it fits, such as encryption, user access levels, backups, data transmission or the energy used by data centres, rather than opinion alone.
- Show balance: give the strongest point on the other side and weigh it against your main argument.
- End with a judgement that answers the question, with a reason: 'Overall ... because ...'.

Code that earns every mark

- Read the requirements twice and tick off each one as your code meets it: every input, rule, output and stored value.
- Handle the edge cases: the first and last items of an array, values exactly on a boundary, equal values when finding a maximum, and the end of a file.
- Keep code clean: meaningful identifiers, consistent indentation, every block closed, and comments only where they help.
- Don't hard-code values that the question says can change; use a constant or a parameter instead.
- Trace your finished code with a boundary value before moving on.

How to use this website

- Build Custom papers at Hard and Extreme, on single subtopics to fix weaknesses and across all topics to test yourself.
- Sit the Mock exams, timed, at least twice. After each one, go to Progress and redo the questions you dropped marks on.
- Use the grade-by-grade checklists for grades 8 and 9: anything you can't tick goes to the top of your list.
- Read the notes and cheatsheets for the less common subtopics, and use the worksheets to prove you really know them.
- Do the Daily 5 every day to stop old weaknesses creeping back, and use Flashcards to keep definitions word-perfect.
- Use the Whiteboard for tracing complex code and drawing logic diagrams.
- For detailed feedback on your extended response answers and Section B code, try 1-to-1 tutoring with Chhetri Academy, starting with the free 30-minute trial lesson.

Timing strategy in the exam

- Aim to finish each paper with about 10 minutes to spare, and use them to trace your code, convert calculations back, and re-read every explain answer for a missing reason.
- Don't over-write short questions: a 2-mark answer needs two precise points, not a paragraph. Save the time for Section B and the extended response question.
- Paper 2: move through Section A briskly so Section B gets your full attention, with at least a minute per mark left for it.

Traps at grade 9

- Dropping marks on easy questions through overconfidence: missing 'two', or the unit asked for.
- Answering the question you expected rather than the one printed, especially on familiar extended response topics.
- Slips in trace tables with nested loops, and mixing up MOD and DIV.
- Truth tables with missing rows or rows out of order.
- Code that works for normal data but fails at a boundary.
- Writing so much on early questions that the last ones are rushed.

Free notes, worksheets, practice papers and mock exams at chhetriacademy.com. Written for this site; not taken from past papers.
Stuck? 1-to-1 tutoring with a free 30-minute trial lesson.