

How to get a grade 9 in Edexcel GCSE Computer Science

A grade 9 needs near-complete accuracy on both papers. You drop almost no marks on knowledge and calculation questions, your Discuss answers reach the top level, and in Paper 2 you finish the hardest programming tasks with code that is correct, robust, efficient and easy to read. Between an 8 and a 9 the difference is usually small slips and time, not knowledge.

What a grade 9 asks of you

- Complete, exact knowledge of every specification statement in Topics 1 to 5, including the less common ones: embedded systems, audit trails and code reviews, utility software, the limitations of binary representation, and ways of identifying network vulnerabilities.
- Flawless working by hand: two's complement arithmetic and overflow, shifts, hexadecimal, full expressions, long trace tables and merge sorts.
- Top-level Discuss answers: detailed, accurate, applied to every part of the scenario, and weighing one side against the other.
- Paper 2: every task finished and tested, including the longest one, with the problem broken into functions, every input validated, files handled correctly and the output exactly as specified.
- The ability to judge efficiency: comparing algorithms and programs by the number of comparisons, the passes through a loop and the use of memory.

Secure the whole specification

- Go through the specification statement by statement. Turn each one into a question and answer it without notes; anything you hesitate on goes on your list.
- Hard spots to check: why computers use binary; how the number of bits limits range and precision; arithmetic shifts on negative numbers; how overflow happens; the stored program concept; how an operating system manages files, processes, peripherals and users.
- Networks: the TCP/IP layers and their protocols, the factors that affect network performance, and penetration testing, ethical hacking, analysis tools and policy reviews as ways to find vulnerabilities.
- Issues and impact with nuance: ownership of and consent for personal data, accountability when an AI system makes a decision, bias in training data, and choosing between open-source and proprietary licences.
- Programming: efficient code, sensible choice of data structure, two-dimensional lists, file processing, and subprograms with clear parameters and return values.

The questions that separate an 8 from a 9

- The longest Paper 2 task: a complete program from a multi-part set of requirements, tested and readable.
- Discuss answers that stay in the top level all the way through, covering every aspect the question names.
- Multi-step expressions and two's complement or overflow questions where one slip loses the mark.
- Writing an efficient algorithm on paper, and explaining why one algorithm is more efficient than another for the data given.
- Finding logic errors in unfamiliar code quickly and precisely.

Programming to a top standard

- Every week, write a full program to a set of requirements against the clock, and aim to finish with time to test.
- Decompose first: write the function headers and the main sequence of calls, then fill each function in and test it on its own.
- Make it robust: validate every input, handle an empty list or file, and avoid crashes on bad data, for example by checking `isdigit()` before using `int()`.
- Make it readable: meaningful names, constants in capitals, comments that explain purpose, consistent indentation and white space.
- Make it efficient: stop a search as soon as the item is found, read a file once rather than inside a loop, and don't repeat work a single loop can do.
- Know the PLS inside out, including `format()` placeholders, string and list methods, file methods, the modules it lists, such as `random` and `math`, and `turtle` graphics.

A 10-week plan

- Aim for 5 to 6 hours a week, plus a Daily 5 every day.
- Weeks 1 and 2: full timed mocks of both papers; use Progress to find every subtopic where you drop marks; try Extreme custom papers on your strongest topics to find hidden gaps.
- Weeks 3 to 5: work through every specification statement; Hard and Extreme custom papers on mixed subtopics; two timed programs a week.
- Weeks 6 to 8: a full timed mock every week, alternating papers; an error log after each; Discuss answers marked honestly against the levels.
- Weeks 9 and 10: redo every question you dropped marks on; a short program every day; cheatsheets and flashcards for exact wording.

Using GCSE Paper Builder

- Custom paper: Hard and Extreme difficulty across mixed subtopics, at full length, marked strictly with the mark scheme.
- Mock exams: sit them under strict timing and aim to finish with time spare, because checking is where a grade 9 is won.
- Progress: chase every subtopic where you still lose marks, however few, and redo those questions until you get full marks.
- Grade-by-grade skill lists: tick off every skill at grades 8 and 9 as well as the ones below.
- Daily 5 and Flashcards: keep the whole course fresh and your definitions exact.
- Whiteboard: plan the structure of long programs and set out trace tables and merge sorts clearly.
- 1-to-1 tutoring with Chhetri Academy (free 30-minute trial lesson) can give you expert feedback on top-level. Discuss answers and on how you design programs.

Timing in the exam

- Paper 1: aim to finish with about 10 minutes to check. Re-check every conversion, expression and trace table, because those are where slips cost marks.
- Paper 2: move briskly through the earlier tasks but test each one before moving on.
- Start the longest task with enough time to plan, write and test it properly. Plan its functions for a couple of minutes before you type.
- Keep 10 minutes at the end to run every file with fresh test data and confirm each is saved under the right name.

Traps at grade 9

- Careless slips: a miscounted bit, a missing $\div 8$ in an expression, a skipped row in a trace table.
- Over-complicating: using Python beyond the PLS that brings new bugs, or writing far more than the question needs.
- Code that works for the example but fails on boundary values, an empty input or the last line of a file.
- Discuss answers that are long but generic, or that ignore one of the aspects the question names.
- Missing a small requirement, such as a message, a rounding rule or the order of the output. Re-read the requirements once your program is finished.

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.