

How to get a grade 7 in OCR GCSE Computer Science

A grade 7 needs precise, well-explained answers throughout Paper 1 and confident programming on Paper 2. The difference from a 5 is in the detail: every explanation has a clear reason linked to the scenario, calculations are done without slips, and in Section B you write complete, working code rather than fragments.

What a grade 7 asks of you

- Explain, not just state: say what happens and why, with the correct technical terms, in the context of the question.
- Compare properly: storage by capacity, speed, portability, durability, reliability and cost; compilers with interpreters; star with mesh; lossy with lossless compression.
- Get every calculation right, including sound file sizes, binary shifts and their effect, overflow, and how many bits are needed.
- Write complete programs in Section B using arrays, string handling, file handling, and functions with parameters and return values.
- Reach the top level of the extended response question most of the time: several developed points, applied to the scenario, looking at more than one side, with a conclusion when asked.

Secure these topics first

- Everything in the grade 5 plan, done accurately.
- 1.1.2 CPU performance: the effect of clock speed, cache size and number of cores, including why more cores don't always make a program run faster.
- 1.2.1 and 1.2.5: why computers need virtual memory and why using it slows the computer down; when to use lossy and when to use lossless compression.
- 1.3.2: IP addresses compared with MAC addresses, what each protocol does, what layers are and why they are used, and how encryption protects data sent over a network.
- 1.4: how each attack works (malware, phishing and other social engineering, brute-force attacks, denial of service, data interception and theft, SQL injection), which method prevents it, and why.
- 2.2.3: 1D and 2D arrays, records, SQL with SELECT, FROM and WHERE, string manipulation, file handling, and sub programs.
- 2.1.3: the stages of merge sort and insertion sort, and how the searching and sorting algorithms compare.
- 2.3.1 and 2.4: defensive design, and drawing logic diagrams from Boolean expressions.

Question types to master

- 4- to 6-mark explain questions set in a scenario.
- Compare questions where every point covers both things.
- The extended response discuss question, written under timed conditions.
- Section B: writing a function, validating input, processing an array or a file, and refining given code.
- Finding, explaining and correcting syntax and logic errors in given code.
- Test plans with specific expected results for normal, boundary, invalid and erroneous data.
- Trace tables with selection inside loops, or with string handling.

Revision plan, week by week

- This plan assumes about 10 weeks. If you have less, shorten weeks 1 to 4 but keep the mocks and reviews.
- Weeks 1 to 4: pair one Paper 1 topic with one Paper 2 topic each week: 1.1 with 2.2; 1.2 with 2.1; 1.3 with 2.3; then 1.4 and 1.5 with 2.4 and 2.5.
- Week 5: 1.6 and extended writing. Write three extended response answers on different scenarios and mark them against a levels mark scheme.
- Weeks 6 and 7: mixed practice on your weakest subtopics, plus two Section B style programs each week, written by hand and then tested on a computer.
- Week 8: a timed mock of each paper, then a full review of every mark you lost.
- Week 9: work on the five weakest subtopics from that review, and redo the questions you dropped marks on.
- Week 10: a second timed mock of each paper, then light review: cheatsheets, flashcards and your list of mistakes.

A programming routine

- Three times a week, write a short program by hand in ERL or your high-level language, then type it in and run it to see whether it works.
- Practise the Section B patterns until you can write them without notes: a validation loop, a running total, finding the largest value in an array, counting items that match a condition, reading every line of a file, and a function that returns a value.
- Rewrite a few of your own Python programs in ERL so that the code in the questions never slows you down.
- Test your own code with normal, boundary, invalid and erroneous data every time; this is exactly what test-plan questions ask for.

How to use this website

- Read the notes and cheatsheet for each subtopic, then do the worksheet and check it against the answers to be sure you really understand it.
- Build Custom papers at Medium and Hard for the subtopics you're working on, and a longer mixed paper across a whole paper's topics once a week.
- Use the grade-by-grade checklists for grades 6 and 7: every skill you can't yet tick is a target.
- Sit the Mock exams in weeks 8 and 10, timed, one paper at a time.
- Use Progress to find your weakest subtopics and redo the questions you dropped marks on.
- Do the Daily 5 every day, and use Flashcards for protocols, laws, registers and definitions.
- Use the Whiteboard for trace tables, sorts and logic diagrams.
- If Section B or the extended response question keeps costing you marks, try 1-to-1 tutoring with Chhetri Academy; the first 30-minute lesson is a free trial.

Timing strategy in the exam

- Paper 1: move briskly through the short questions (about a minute per mark) so you have plenty of time for the extended response answer and a few minutes to check.
- Paper 2: use the marks to split your time between Section A and Section B at about a minute per mark, leaving 5 minutes to check. Work out the clock time you must start Section B and stick to it.
- Before writing code, spend 30 seconds listing the steps (input, validate, process, output). It saves crossing out later.

Traps at grade 7

- Describing when the question says explain: every point needs a reason.
- Extended response answers that could be about any organisation. Name the people and the data in the scenario.
- Off-by-one loop errors, and using `<` where `<=` is needed.
- Forgetting to cast an input to a number before doing arithmetic, or forgetting to return a value from a function.
- Counting string positions from 1 instead of 0 when using `.substring`.
- Vague expected results in test plans, such as 'it works' or 'error'.
- Careless binary slips. Always convert your answer back to check it.

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.