

Grade 7 checklist

52 skills. Tick each one when you can do it. For grade 7 you need the skills for the grades below it too.

Computational thinking

- ☐ **Explain benefits in the context given** Link each benefit to the scenario with a clear consequence, e.g. '... so the app is finished sooner'. (Decomposition and abstraction)
- ☐ **Write an algorithm using selection and repetition** For example, a loop that keeps asking for an input until the value is valid. (Flowcharts, pseudocode and program code)
- ☐ **Choose between an array and a record** A record holds related fields of different types; an array holds items of the same type. (Variables, constants and data structures (strings, arrays and records))
- ☐ **Use modulus and integer division to solve problems** E.g. test whether a number is even, or convert minutes into hours and minutes. (Arithmetic, relational and logical operators)
- ☐ **State the purpose of an algorithm** Spot the pattern in the trace, e.g. 'it adds up the digits of a number'. (Tracing algorithms and trace tables)
- ☐ **Correct logic errors in an algorithm** Give the line number, what is wrong and the corrected code. (Syntax, logic and runtime errors)
- ☐ **Compare linear and binary search** Weigh speed on large lists against simplicity and working on unsorted data. (Linear and binary search)
- ☐ **Compare bubble sort and merge sort** Weigh speed on large lists against memory use and simplicity. (Bubble sort and merge sort)
- ☐ **Explain improvements to a bubble sort** Stop after a pass with no swaps, and skip the items already in place. (Bubble sort and merge sort)
- ☐ **Suggest how to make an algorithm more efficient** E.g. stop a loop as soon as the answer is known. (Efficiency of algorithms)
- ☐ **Explain why an algorithm is not fit for purpose** Use test data to show a case where it gives the wrong result. (Efficiency of algorithms)
- ☐ **Write a logic statement from a description** Turn a rule such as 'on if A and not B, or if C' into a statement. (Truth tables and logic)

Data

- ☐ **Work out the minimum bits for a requirement** Find the smallest power of 2 that is at least the number of different values needed. (Binary representation and unsigned binary–denary conversion)
- ☐ **Explain the effect of an overflow** The carried-out bit is lost, so the stored result is wrong (256 less than the true total for 8 bits). (Binary addition and overflow)
- ☐ **Explain loss of precision in a right shift** Bits shifted off the right-hand end are lost, so an answer that should have a fraction is rounded down. (Logical and arithmetic shifts)
- ☐ **Explain overflow in a left shift** A 1 shifted off the left-hand end is lost, so the result is wrong because it is too big for 8 bits. (Logical and arithmetic shifts)

- ☐ **Convert longer binary and hex values** Group the bits in fours from the right: each group is one hex digit, so 16 bits are 4 hex digits. (Hexadecimal)
- ☐ **Explain the limitations of 7-bit ASCII** 128 codes cannot include accented letters, other alphabets or most symbols and emoji. (Characters and ASCII)
- ☐ **Construct a file size expression for an image** width × height × colour depth gives bits; ÷ 8 for bytes, then ÷ 1024 for KiB. (Bitmap images)
- ☐ **Construct a file size expression for sound** sample rate × bit depth × duration in seconds gives the size in bits. (Sound)
- ☐ **Explain precision limits when storing measurements** A fixed number of bits gives fixed steps, so values between the steps have to be rounded. (Limitations of binary representation)
- ☐ **Calculate the total storage for several files** Convert every size to the same unit before adding or multiplying. (Units and file size calculations)
- ☐ **Explain when RLE works well** RLE only saves space when the data has long runs of the same value. (Lossless and lossy compression (incl. RLE))

Computers

- ☐ **Explain how a jump changes sequential execution** Instructions normally run in the order they are stored, but a jump puts a new address in the program counter, so instructions can be repeated or skipped. (Stored program concept and von Neumann architecture)
- ☐ **Describe decoding and executing a given instruction** Say how the CU decodes it, which bus carries what, and what the ALU or memory does with it. (CPU components and the fetch–decode–execute cycle)
- ☐ **Calculate addressable memory from address bus width** An n-bit address bus can address 2^n memory locations. (CPU components and the fetch–decode–execute cycle)
- ☐ **Explain why a feature matters in context** For example, a pacemaker must use very little power because its battery is inside the patient's body. (Embedded systems)
- ☐ **Apply the four functions to a scenario** Work out which function each part of a scenario needs, and describe it using the scenario's details. (Operating systems)
- ☐ **Compare full and incremental backups** Incremental backups are quicker and smaller, but restoring needs the full backup plus every incremental one since. (Utility software)
- ☐ **Spot vulnerabilities in a piece of code** Look for unvalidated input, passwords written into the code and unlimited log-in attempts, and suggest a fix for each. (Robust software: audit trails and code reviews)
- ☐ **Choose a suitable translator for a situation** An interpreter suits writing and testing; a compiler suits a finished program given to users. (Programming languages and translators)

Networks

- ☐ **Discuss whether an organisation should use a network** Weigh the benefits against the costs, security risks and reliance on central devices, then reach a justified conclusion. (LANs, WANs and why computers are connected)

- ☐ **Explain why IPv6 was introduced** 32-bit IPv4 allows only about 4.3 billion addresses, which is not enough for all the devices now online. (The internet: IP addressing and routers)

- ☐ **Recommend a connection type with justification** Match speed, range, latency, mobility, security and cost to the needs of the users in the scenario. (Wired and wireless connectivity and performance)

- ☐ **Rearrange the formula for size or rate** Use $\text{file size} = \text{transmission rate} \times \text{time}$, or $\text{transmission rate} = \text{file size} \div \text{time}$. (Network speeds and transmission time)

- ☐ **Explain why real transfers take longer** Shared bandwidth, interference, packet headers and resent packets all add to the time taken. (Network speeds and transmission time)

- ☐ **Describe the roles of TCP and IP** TCP splits data into numbered packets and makes sure they all arrive in order; IP addresses and routes the packets. (Protocols)

- ☐ **Describe how data passes through the layers** Down the layers on the sender, with each adding a header, then up the layers on the receiver, with each removing one. (The TCP/IP model)

- ☐ **Calculate the cables needed for a full mesh** A full mesh of n devices needs $n(n - 1) \div 2$ cables, e.g. 6 devices need 15. (Network topologies)

- ☐ **Describe the stages of a penetration test** Agree the scope and get permission, gather information, attempt attacks, report, then fix and retest. (Network security and identifying vulnerabilities)

- ☐ **Interpret a table of firewall rules** Apply the rules in order to decide whether each packet is allowed or blocked. (Protecting networks)

Issues and impact

- ☐ **Explain why 'anonymous' data may identify people** Combining details such as postcode, date of birth and location can point to one person. (Personal data: privacy, ownership, consent and data protection)

- ☐ **Explain accountability and legal liability problems** It is hard to say who is to blame, or who must pay, when a machine made the decision itself. (Artificial intelligence, machine learning and robotics)

- ☐ **Explain what a licence allows in context** Work out whether the code may be changed, sold or shared, and whether changes must be released too. (Intellectual property and licensing)

- ☐ **Explain why social engineering beats technical defences** A genuine user hands over access, so firewalls, encryption and passwords are bypassed. (Malware, social engineering and technical vulnerabilities)

- ☐ **Explain the limits of each protection method** For example, encryption does not stop data being deleted or locked by ransomware, so backups are still needed. (Protecting digital systems and data (incl. encryption))

Problem solving with programming

- ☐ **Write validation and authentication code** Repeat an input until it passes every check, and check an ID and password against stored data. (Developing code: readability, errors, validation and testing)

- ☐ **Use nested loops and nested selection** Put a loop or if inside another and work out how many times the inner block runs. (Sequence, selection and iteration)

- ☐ **Use a two-dimensional list** Access an item with `name[row][column]` and use nested loops to process every item. (Data types and data structures)

- ☐ **Code a validation loop for user input** Keep asking until the input passes every check, showing a helpful message each time. (Input, output and file handling)

 - ☐ **Use modulus to test divisibility** $n \% 2 == 0$ tests for an even number; $n \% 10$ gives the last digit.
(Operators)

 - ☐ **Write a function that returns a value** Take parameters, calculate a result, return it and use the returned value in the main program. (Subprograms and libraries)
-

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.