

Grade 6 checklist

62 skills. Tick each one when you can do it. For grade 6 you need the skills for the grades below it too.

Computational thinking

- ☐ **Apply abstraction to a real-world model** Say which details a model must keep and which it can leave out, linked to its purpose. (Decomposition and abstraction)
- ☐ **Compare count-controlled and condition-controlled loops** A count-controlled loop repeats a set number of times; a condition-controlled loop repeats while a condition is True. (Flowcharts, pseudocode and program code)
- ☐ **Use a two-dimensional array** Find or change an item using a row index and then a column index. (Variables, constants and data structures (strings, arrays and records))
- ☐ **Apply operator precedence to evaluate expressions** Brackets first, then exponentiation, then multiplication and division, then addition and subtraction. (Arithmetic, relational and logical operators)
- ☐ **Trace algorithms that use arrays or strings** Use the index on each pass to find the correct item or character. (Tracing algorithms and trace tables)
- ☐ **Find a logic error using test data** Compare the expected result with the actual result, then trace to find the line. (Syntax, logic and runtime errors)
- ☐ **Show the stages of a binary search** Give each middle item checked and the part of the list that is left. (Linear and binary search)
- ☐ **Describe how a merge sort works** Split the list into single items, then merge pairs of sorted lists until one list is left. (Bubble sort and merge sort)
- ☐ **Show the stages of a merge sort** Write the lists after each split and after each round of merging. (Bubble sort and merge sort)
- ☐ **Find maximum comparisons for a binary search** Find the smallest power of 2 that is greater than the number of items. (Efficiency of algorithms)
- ☐ **Use working columns for brackets and NOT** Work out each part of the statement in its own column before the output. (Truth tables and logic)

Data

- ☐ **Find the largest unsigned value for n bits** The largest value is $2^n - 1$, because one of the patterns is used for 0. (Binary representation and unsigned binary–denary conversion)
- ☐ **Convert two's complement binary to denary** Treat the MSB as -128 and add the place values of the other 1s. (Two's complement)
- ☐ **Convert negative denary to two's complement** Write the positive number in 8 bits, invert every bit, then add 1. (Two's complement)
- ☐ **Spot an overflow in an 8-bit addition** A 1 carried out of the most significant bit means the result needs 9 bits. (Binary addition and overflow)

- ☐ **Apply an arithmetic shift right** Move every bit right and fill the gaps on the left with copies of the sign bit (the MSB). (Logical and arithmetic shifts)

- ☐ **Explain why hexadecimal is used** It is shorter and easier for people to read, remember and copy than binary, and it converts to binary easily. (Hexadecimal)

- ☐ **Convert between binary codes and characters** Convert the 7-bit code to denary, then count on or back from a code you know. (Characters and ASCII)

- ☐ **Explain effects of changing resolution or colour depth** More pixels or more bits per pixel improve quality but increase the file size. (Bitmap images)

- ☐ **Calculate the sample interval** Sample interval = $1 \div \text{sample rate}$: the time between one sample and the next. (Sound)

- ☐ **Explain effects of sample rate and bit depth** Increasing either makes the recording a closer copy of the original, but makes the file larger. (Sound)

- ☐ **Explain the limits of colour and bit depth** Few bits give few colours or amplitude values, so detail is lost or values are rounded. (Limitations of binary representation)

- ☐ **Explain the limits of a character set** 7-bit ASCII has only 128 codes, so it cannot represent every character in every language. (Limitations of binary representation)

- ☐ **Explain why storage uses multiples of 1024** $1024 = 2^{10}$, a power of 2, which suits binary computers. (Units and file size calculations)

- ☐ **Construct an expression for a file size** Build it stage by stage: size in bits, then $\div 8$, then $\div 1024$ for each unit. (Units and file size calculations)

- ☐ **Encode and decode data using RLE** Replace each run of a repeated value with a count and the value, e.g. 5A. (Lossless and lossy compression (incl. RLE))

Computers

- ☐ **Explain why new tasks need no rewiring** A different program is simply loaded into memory, and the CPU executes whatever instructions are stored there. (Stored program concept and von Neumann architecture)

- ☐ **Describe the fetch stage step by step** Address out on the address bus, read signal on the control bus, instruction back on the data bus, program counter incremented. (CPU components and the fetch–decode–execute cycle)

- ☐ **Describe how solid-state storage works** Flash memory cells trap an electrical charge (or not) to store 1s and 0s, with no moving parts. (Main memory and secondary storage)

- ☐ **Compare storage by capacity, speed, cost, durability** For example, an SSD is faster and tougher than a hard disk drive but costs more per GiB. (Main memory and secondary storage)

- ☐ **Compare embedded and general-purpose computers** One task with a fixed program, compared with many tasks and software the user can install. (Embedded systems)

- ☐ **Explain how multitasking works on one processor** The OS gives each process a short time slice in turn, switching so fast that they seem to run at the same time. (Operating systems)

- ☐ **Explain how defragmentation speeds up a hard disk** It puts the fragments of each file back together, so the read/write head moves less. (Utility software)

- ☐ **Explain how audit trails help find problems** They show which change caused a fault, or reveal unusual activity that suggests an attack. (Robust software: audit trails and code reviews)

- ☐ **Describe how a compiler translates code** It translates the whole program into machine code before it runs, producing an executable file. (Programming languages and translators)

- ☐ **Describe how an interpreter translates code** It translates and runs one line at a time, every time the program runs, and stops at the first error. (Programming languages and translators)

Networks

- ☐ **Explain benefits of a network in context** Link each benefit to the scenario, e.g. records stored on a server can be opened from every computer in the building. (LANs, WANs and why computers are connected)

- ☐ **Describe how routers forward packets** Each router reads a packet's destination IP address and sends the packet on towards the destination network. (The internet: IP addressing and routers)

- ☐ **Convert an IPv4 address to binary** Convert each of the four numbers to an 8-bit binary number, keeping any leading zeros. (The internet: IP addressing and routers)

- ☐ **Compare IPv4 and IPv6 addresses** IPv4 uses 32 bits written as four denary numbers; IPv6 uses 128 bits written as groups of hexadecimal digits. (The internet: IP addressing and routers)

- ☐ **Explain why Wi-Fi performance falls** Distance, walls, interference and sharing the bandwidth between many devices all reduce performance. (Wired and wireless connectivity and performance)

- ☐ **Construct an expression for transmission time** For example, $20 \times 1024 \times 1024 \times 8 \div (40 \times 1,000,000)$ seconds for a 20 MiB file at 40 Mbps. (Network speeds and transmission time)

- ☐ **Compare POP3 and IMAP** POP3 downloads emails to one device; IMAP keeps them on the server so that every device shows the same emails. (Protocols)

- ☐ **Explain benefits of a layered model** Layers are independent, so one can change without affecting the others, and standards let products from different makers work together. (The TCP/IP model)

- ☐ **Compare the reliability of the topologies** Explain what stops working when a cable or a central device fails in each topology. (Network topologies)

- ☐ **Explain why organisations use penetration testing** To find vulnerabilities before criminals exploit them, so that they can be fixed. (Network security and identifying vulnerabilities)

- ☐ **Distinguish ethical hacking from malicious hacking** Ethical hackers have permission and report what they find; malicious hackers have no permission and aim to cause harm. (Network security and identifying vulnerabilities)

- ☐ **Explain how a firewall works** It checks each packet against rules (e.g. IP addresses and port numbers) and blocks packets that break them. (Protecting networks)

Issues and impact

- ☐ **Explain the effect of short replacement cycles** Replacing devices more often means more are manufactured (materials, energy, emissions) and more become e-waste. (Environmental issues)

- ☐ **Explain misuse and the effects of breaches** Misuse and breaches can lead to fraud or identity theft for people, and fines and lost trust for organisations. (Personal data: privacy, ownership, consent and data protection)

- ☐ **Explain safety risks of autonomous machines** A self-driving car or robot that makes a wrong decision, meets a new situation or is hacked can injure people. (Artificial intelligence, machine learning and robotics)

- ☐ **Explain pros and cons of open source** Usually free, customisable and community-supported, but may have no official support or guaranteed updates. (Intellectual property and licensing)

- ☐ **Explain how unpatched software is exploited** Attackers use known, published security flaws that have not been fixed on that computer. (Malware, social engineering and technical vulnerabilities)

- ☐ **Spot and explain signs of phishing** Urgency, an unexpected request, a generic greeting and a link to the wrong web address. (Malware, social engineering and technical vulnerabilities)

- ☐ **Describe a backup and recovery procedure** Regular automatic backups kept off site and offline, several versions, tested restores and a recovery plan. (Protecting digital systems and data (incl. encryption))

Problem solving with programming

- ☐ **Locate and correct errors in a program** Find the line with the error, name the type of error and write the corrected line. (Developing code: readability, errors, validation and testing)

- ☐ **Convert a flowchart or pseudocode into Python** Turn each input, process, decision and output of an algorithm into the matching Python statement. (Developing code: readability, errors, validation and testing)

- ☐ **Iterate over every item of a data structure** Use for item in a list, or for character in a string, to process each one in turn. (Sequence, selection and iteration)

- ☐ **Find length, position, substrings and case** Use len(), find(), slicing such as word[1:4], upper() and lower(). (Data types and data structures)

- ☐ **Write records to a text file** Build each record as one string with commas and a newline, then close the file. (Input, output and file handling)

- ☐ **Apply the order of operations** Brackets first, then exponentiation, then *, /, // and %, then + and -. (Operators)

- ☐ **Write a procedure that takes parameters** Define it with def, pass values in when you call it, and let it carry out its task. (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.