

Programme of study for Year 9 Computer Science and IT

Autumn (1 st term) Topic	Autumn (2 nd term) Topic	Spring (1 st term) Topic	Spring (2 nd term) Topic	Summer (1 st term) Topic	Summer (2 nd term) Topic
Systems Architecture (6) Exploring how a computer's central processing unit is structured and how it carries out instructions – building the hardware foundation that underpins every other topic in Year 9.	Data Representation: Hexadecimal and Beyond (6) Levelling up from Year 8's binary foundations – introducing hexadecimal, negative number representation, Unicode and compression.	Cybersecurity: Attack Vectors and Vulnerabilities (6) A full technical picture of how systems are attacked – from human-targeted social engineering to technical exploits – and the countermeasures that protect against them.	Python: Consolidation and Extension (6) Consolidating Year 8's Python foundations and introducing the accessible new concepts – casting, string manipulation, lists and functions – that directly enable algorithm implementation in Summer 1.	Computational Thinking and Algorithms (6) Developing the structured problem-solving mindset of computer scientists, then applying it to real searching and sorting algorithms – traced on paper and implemented in Python.	App Design and Build – AppShed (6) Applying design thinking, iterative development and evaluation skills to plan, build, test and present a real mobile app – the Year 9 capstone project.
Skills: Identifying and describing the components of the Von Neumann architecture. Explaining the roles of the ALU, Control Unit, registers and buses. Tracing the Fetch-Decode-Execute cycle. Explaining how cores, clock speed and cache size affect CPU performance. Distinguishing RAM from ROM. Comparing types of secondary storage.	Skills: Converting fluently between hexadecimal, binary and denary. Applying two's complement to represent negative numbers. Distinguishing Unicode from ASCII and explaining why Unicode was needed. Distinguishing lossless from lossy compression. Applying run-length encoding.	Skills: Identifying and explaining social engineering attack types. Distinguishing categories of malware. Explaining technical attack methods including brute force, denial of service and SQL injection. Recognising network interception threats. Evaluating the effectiveness of security countermeasures.	Skills: Applying type casting to convert between data types. Manipulating strings using built-in methods and slicing. Creating and traversing lists. Combining lists with selection and iteration in programs. Writing and calling simple functions with parameters.	Skills: Applying decomposition, abstraction and pattern recognition to real problems. Writing and reading pseudocode. Constructing and tracing flowcharts using standard symbols. Tracing and comparing linear and binary search. Tracing bubble sort. Implementing searching and sorting algorithms in Python.	Skills: Identifying design principles for effective mobile apps. Planning an app for a specific audience and purpose using wireframes. Building a multi-screen app using AppShed. Testing systematically against design criteria. Evaluating and refining an app based on structured feedback.
Key Learning Outcomes: Describe the Von Neumann architecture and identify its key components. Explain the	Key Learning Outcomes: Convert numbers between hexadecimal, binary and denary in all directions. Use	Key Learning Outcomes: Explain phishing, pretexting, baiting and shouldering as social engineering methods.	Key Learning Outcomes: Use int(), float() and str() to convert between data types correctly. Apply string methods	Key Learning Outcomes: Apply decomposition, abstraction and pattern recognition to break down and	Key Learning Outcomes: Identify and apply key

Autumn (1 st term) Topic	Autumn (2 nd term) Topic	Spring (1 st term) Topic	Spring (2 nd term) Topic	Summer (1 st term) Topic	Summer (2 nd term) Topic
roles of the ALU, Control Unit, registers and buses within the CPU. Trace the Fetch-Decode-Execute cycle step by step, naming what happens at each stage. Explain how the number of cores, clock speed and cache size each affect CPU performance. Distinguish between RAM and ROM and explain the role of each. Compare magnetic, optical and solid-state secondary storage, evaluating the trade-offs between them.	two's complement to represent and calculate with negative binary numbers. Explain why Unicode was developed to replace ASCII and describe the difference between them. Distinguish lossless compression (no data lost, e.g. run-length encoding) from lossy compression (some data discarded, e.g. JPEG). Apply run-length encoding to a simple data set and calculate the resulting saving.	Distinguish between and explain viruses, worms, ransomware, spyware and trojans. Describe how brute force, denial of service and SQL injection attacks work. Explain packet sniffing and man-in-the-middle attacks. Evaluate security countermeasures including firewalls, anti-malware, encryption, access levels, biometrics and penetration testing.	(.upper(), .lower(), .find(), len()) and slicing to manipulate text. Create, index, traverse and modify lists using append(), len() and remove(). Build programs that combine lists with for loops and if statements to solve problems. Write simple functions with parameters, call them from a program, and explain why functions make programs easier to maintain.	simplify problems. Write and trace algorithms expressed in pseudocode. Construct flowcharts using standard symbols and trace them to predict output. Trace a linear search and a binary search, and compare their efficiency. Trace a bubble sort, explaining what happens at each pass. Implement a linear search and a bubble sort in Python, using a list and a for loop.	principles of effective mobile app design (navigation, readability, consistency, accessibility). Plan an app for a given audience and purpose, producing labelled wireframes for all screens. Build a multi-screen, interactive app in AppShed that meets the original brief. Test the app systematically using a structured test plan. Evaluate the finished app against the original design criteria and identify specific improvements.
Half Term 1 Evidence to cover: CPU component labelling and explanation task; traced Fetch-Decode-Execute cycle; written comparison of secondary storage types.	Half Term 2 Evidence to cover: Hexadecimal conversion exercises (all three directions); two's complement calculation task; run-length encoding applied task with compression ratio calculated.	Half Term 3 Evidence to cover: Attack type classification and explanation task; countermeasures evaluation scenario; extended written response linking attack vectors to appropriate defences.	Half Term 4 Evidence to cover: A series of short Python programs demonstrating casting, string manipulation, list operations and a simple function; end-of-unit programming task combining all four concepts.	Half Term 5 Evidence to cover: Pseudocode and flowchart construction tasks; linear search and binary search trace exercises with comparison; bubble sort trace; Python implementation of linear search and bubble sort.	Half Term 6 Evidence to cover: Completed app project including: wireframe designs, built and published AppShed app, structured test plan with results,

Autumn (1 st term) Topic	Autumn (2 nd term) Topic	Spring (1 st term) Topic	Spring (2 nd term) Topic	Summer (1 st term) Topic	Summer (2 nd term) Topic
					written evaluation against design criteria.
Rationale for sequence: Year 9 opens by establishing exactly how a computer works at hardware level – knowledge that makes every subsequent unit more meaningful. Understanding CPU architecture and memory gives students the technical context to appreciate why data must be represented efficiently (Autumn 2), why systems can be attacked (Spring 1), and why algorithmic efficiency matters (Summer 1). It is also the single biggest GCSE topic that has had no prior KS3 coverage.	Rationale for sequence: Hexadecimal connects directly to Year 8 binary work and to the CSS colour codes students used in their webpages – making it immediately recognisable. Placing this unit second means students understand how data is represented in the CPU they studied in Autumn 1. Two's complement deepens number representation beyond what is possible at KS3. Compression is introduced here as a logical extension of the image and file size concepts from Year 8 Spring 2.	Rationale for sequence: Placing this unit third means students now understand the systems being attacked (Autumn 1) and how data is stored within them (Autumn 2), so technical exploits like SQL injection and packet sniffing have genuine technical meaning. This unit deliberately extends Year 8's data protection and ethical hacking content, pivoting from legal awareness and encryption principles to a full technical taxonomy of attack vectors and vulnerabilities in line with OCR J277 section 1.4.	Rationale for sequence: Python is placed before the Algorithms unit deliberately: when students trace and implement a linear search or bubble sort in Summer 1, they need to know how to traverse a list in Python. Teaching lists here means algorithm implementation is a natural extension rather than a double learning challenge. Casting is prioritised first because it resolves the most common source of runtime errors in student programs. Functions are introduced at concept level only – return values and scope are reserved for Year 10 where there is more time to explore them properly.	Rationale for sequence: Placing this unit after Python ensures students can implement algorithms in code rather than just trace them on paper – a far more powerful form of understanding. The list traversal skills from Spring 2 map directly onto linear search and bubble sort. Placing this in Summer 1 rather than Spring 2 also means students arrive at the algorithm unit with stronger Python fluency, so the focus can be on algorithmic thinking rather than syntax.	Rationale for sequence: App Design and Build closes Year 9 as a capstone project that draws on skills developed across the whole year – the iterative design process mirrors the systematic problem-solving of the Algorithms unit; the audience-and-purpose thinking echoes the web design work from Year 8; and the evaluation criteria connect to the testing vocabulary from across both years. Placing an applied, creative project at the end of the year provides a motivating finale and produces something students can genuinely showcase.

Autumn (1 st term) Topic	Autumn (2 nd term) Topic	Spring (1 st term) Topic	Spring (2 nd term) Topic	Summer (1 st term) Topic	Summer (2 nd term) Topic
Home – Learning: Research the Fetch-Decode-Execute cycle and write a plain-English explanation of each stage, as if explaining it to someone who has never studied Computer Science.	Home – Learning: Find the hex colour codes for three colours used on a website you designed in Year 8 and convert each one into its binary and denary equivalents. Show your working.	Home – Learning: Research one real-world cyberattack (e.g. WannaCry, the TalkTalk breach, the NHS ransomware attack). Summarise what type of attack it was, how it worked, and what countermeasures could have prevented it.	Home – Learning: Complete two short Python exercises on a platform of your choice: one using string manipulation, and one using a list. Bring a printout or screenshot to the next lesson.	Home – Learning: Trace a bubble sort on the list [5, 2, 8, 1, 9, 3], showing every comparison and swap across every pass until the list is fully sorted.	Home – Learning: Download and review one app on a device at home. Write down three specific design choices that show it was designed with its target audience in mind, and one thing you would improve.
Reading / High Quality Text: Systems Architecture – BBC Bitesize GCSE Computer Science https://www.bbc.co.uk/bitesize/guides/z2342hv/revision/1 Literacy: Technical terms: Von Neumann, ALU, Control Unit, register, bus, FDE cycle, cache, core, RAM, ROM, volatile, non-volatile.	Reading / High Quality Text: Hexadecimal – BBC Bitesize GCSE Computer Science https://www.bbc.co.uk/bitesize/guides/zp73wmn/revision/1 Literacy: Technical terms: hexadecimal, two's complement, Unicode, ASCII, lossless, lossy, compression ratio, run-length encoding.	Reading / High Quality Text: Cyber Security – BBC Bitesize GCSE Computer Science https://www.bbc.co.uk/bitesize/guides/zp3bkqt/revision/1 NCSC Cyber Security Advice for Young People https://www.ncsc.gov.uk/section/information-for/individuals-families Literacy: Technical terms: social engineering, phishing, pretexting, malware, ransomware, brute force, SQL injection, denial of service, packet sniffing, man-in-the-middle, firewall, biometrics.	Reading / High Quality Text: Python Lists – W3Schools https://www.w3schools.com/python/python_lists.asp Python String Methods – W3Schools https://www.w3schools.com/python/python_strings_methods.asp Literacy: Technical terms: casting, type conversion, string, index, slice, list, append, function, parameter, argument.	Reading / High Quality Text: Algorithms – BBC Bitesize GCSE Computer Science https://www.bbc.co.uk/bitesize/guides/z22wwmn/revision/1 Literacy: Technical terms: decomposition, abstraction, pattern recognition, pseudocode, flowchart, algorithm, linear search, binary search, bubble sort, pass, iteration, efficiency.	Reading / High Quality Text: AppShed Getting Started Guide https://appshed.com Mobile App Design Principles – Google Material Design https://m3.material.io/foundations Literacy: Technical terms: wireframe, prototype, navigation, usability, accessibility, iteration, test plan, evaluation, design criteria, user interface.
Numeracy: Calculating theoretical instructions per second from	Numeracy: Multi-step conversion calculations between	Numeracy: Statistics on cybercrime frequency and financial	Numeracy: Using lists to store and process numerical data (e.g. calculating	Numeracy: Counting comparisons and swaps in search and sort	Numeracy: Screen dimensions and

Autumn (1 st term) Topic	Autumn (2 nd term) Topic	Spring (1 st term) Topic	Spring (2 nd term) Topic	Summer (1 st term) Topic	Summer (2 nd term) Topic
clock speed; comparing cache sizes in KB/MB; evaluating performance trade-offs numerically.	hex/binary/denary; two's complement arithmetic; calculating compression ratios and percentage savings.	impact; calculating brute force attempt counts from password complexity; evaluating time-to-crack estimates.	totals and averages from a list); type conversion between integer and float values.	algorithms; understanding that binary search requires a sorted list; comparing the number of steps needed for linear vs binary search on different list sizes.	aspect ratios for mobile devices; proportional layout decisions in app design; quantitative test plan recording (pass/fail counts, error frequency).
Enrichment / opportunities to develop cultural capital (including careers, WRL and SMSC): Participate in CyberFirst activities and the Bebras Computational Thinking Challenge. Explore careers such as Systems Engineer, Cybersecurity Analyst, Software Developer, Algorithm Engineer and UX Designer. Attend virtual careers talks and computing competitions. Link to OCR J277 GCSE Computer Science preparation.					