Key Stage 4 curriculum vision

This curriculum transitions students from digital creators to technical computer scientists. Aligned with the OCR J277 specification, the curriculum integrates practical programming alongside theoretical components 1 and 2 across the full two-year cycle.

Year 10: Introduces fundamental coding concepts and the architectural principles of computer systems, establishing a strong technical baseline.

Year 11: Elevates these core principles, challenging students to apply their ongoing coding and theoretical knowledge to complex programming projects and rigorous computational thinking.

Year 10 overview

Topic Title	Unit 1.1 Architecture of the CPU
Pupils will know	• The purpose of the CPU and the mechanics of the Fetch-Decode-Execute (FDE) cycle. • The core components of the Von Neumann architecture: the Control Unit (CU), Arithmetic Logic Unit (ALU), and Cache. • The role of internal CPU registers: Program Counter (PC), Memory Address Register (MAR), Memory Data Register (MDR), and Accumulator (ACC). • How clock speed, cache size, and the number of processor cores dynamically affect CPU performance. • The purpose and characteristics of embedded systems vs. general-purpose computers, including real-world examples.
Pupils will know how to (procedural knowledge)	• Demonstrate how the FDE cycle moves instructions and data around through the various registers. • Identify embedded systems within modern appliances. • Evaluate a set of hardware specifications to determine which system will perform faster under specific workloads.
Core knowledge to know and remember to progress to later learning (foundational knowledge)	• How hardware physically stores and processes data using binary representation
Prior knowledge	Year 8 Hardware & Networking
Future knowledge	**Linked Units**: Y10 Unit 1.2 Memory and Storage (how registers fetch data from RAM) and A Level Unit 1.1 Processors (advanced registers, pipelining, CISC/RISC architectures).
Key vocabulary	Von Neumann, FDE Cycle, Register, Program Counter, MAR, MDR, Accumulator, Control Unit, ALU, Cache, Clock Speed, Cores, Embedded

	System.

Topic Title	1.2 Data Representation & Storage
Pupils will know	• The difference between volatile memory (RAM) and non-volatile memory (ROM), alongside the purpose and functionality of virtual memory. • The necessity for secondary storage in computer systems. • The unique characteristics, advantages, and disadvantages of three storage types: Magnetic, Optical, and Solid-State. • Data units and sizes: Bit, Nibble, Byte, Kilobyte, Megabyte, Gigabyte, Terabyte, Petabyte. • The mathematical processes for converting numbers up to 8 bits between Binary, Denary, and Hexadecimal formats. • The rules of binary addition, overflow errors, and binary shifts (multiplication/division). • How characters (ASCII/Unicode), images (pixels, metadata, color depth, resolution), and sound (sample rate, bit depth) are represented in binary. • The difference between lossy and lossless compression.
Pupils will know how to (procedural knowledge)	• Convert numbers up to 8 bits between denary, binary, and hexadecimal. • Perform binary additions and arithmetic shifts (left/right) to multiply or divide values by powers of two. • Calculate the storage requirements for text, image, and sound files using standard data units.
Core knowledge to know and remember to progress to later learning (foundational knowledge)	• Knowing that all digital data, regardless of media type, is ultimately represented as combinations of binary states.
Prior knowledge	KS3 Thread: Computer Science / Information Technology (Binary foundations and digital media metrics taught in Year 9 Data representation).
Future knowledge	**Linked Units:** Y11 Unit 2.2 Programming Fundamentals (how data types declare storage sizes in RAM) and A Level Unit 1.4 Data Types (Two's Complement signed numbers, Floating Point normalization).
Key vocabulary	Volatile, Non-Volatile, RAM, ROM, Virtual Memory, Secondary Storage, Magnetic, Solid-State, Optical, Binary, Hexadecimal, Overflow, Shift, ASCII, Unicode, Color Depth, Metadata, Compression.

Topic Title	1.5 Systems Software
Pupils will know	• The purpose and core functions of an Operating System (OS). • How the OS handles user interfaces (CLI, GUI), memory management, multitasking, peripheral management, file management, and user access/security. • The purpose and utility of standard utility software packages. • The specific functions of defragmentation software and compression software.
Pupils will know how to (procedural knowledge)	• Explain the roles of the OS managers • Identify utility software and explain its role.
Core knowledge to know and remember to progress to later learning (foundational knowledge)	• How system software acts as the interface between complex low-level physical hardware and the user.
Prior knowledge	Y10 Unit 1.1 Systems Architecture and Unit 1.2 Memory and Storage (understanding what hardware resources the OS is actively managing).
Future knowledge	**Linked Units:** Y11 Unit 2.5 Languages & IDEs (how software interacts with translators) and A Level Unit 1.2 Software & Development (scheduling algorithms like Round Robin, virtual machines, and memory paging/segmentation).
Key vocabulary	Operating System, User Interface, Memory Management, Multitasking, Peripherals, Utility Software, Defragmentation, Compression.

Topic Title	2.1 Algorithms
Pupils will know	• The core computational thinking principles: Abstraction (removing unnecessary details) and Decomposition (breaking problems down). • The concept of algorithmic thinking and the structural requirement for precise steps. • The logical mechanisms behind search routines: Linear Search and Binary Search (and the prerequisite for sorted datasets). • The structural mechanics of sort routines: Bubble Sort, Merge Sort, and Insertion Sort.
Pupils will know how to	• Interpret, complete, correct, and write formal algorithms using

(procedural knowledge)	standard visual flowcharts. ● Read, evaluate, and write structured algorithm logic using pseudocode and OCR Exam Reference Language (ERL). ● Manually dry-run data arrays through search and sort algorithms using strict variable trace tables.
Core knowledge to know and remember to progress to later learning (foundational knowledge)	● Recognising recurring structural patterns across algorithms to evaluate logic paths independent of specific language syntax rules.
Prior knowledge	KS3 Thread: Computer Science (Algorithm mapping and block-based/introductory textual logic from Year 8 and 9).
Future knowledge	**Linked Units**: Y11 Unit 2.2 Programming Fundamentals (converting pseudocode designs into working projects) and A Level Unit 2.3 Algorithms to Solve Problems (advanced sorting mechanics like Quick Sort, pathfinding, and algorithmic complexity proofs).
Key vocabulary	Abstraction, Decomposition, Algorithmic Thinking, Flowchart, Pseudocode, OCR ERL, Linear Search, Binary Search, Bubble Sort, Merge Sort, Insertion Sort, Trace Table, Pass.

Topic Title	**2.2 Programming Fundamentals**
Pupils will know	● The application of the three core programming constructs: Sequence, Selection (IF/ELSE structures), and Iteration (Count-controlled loops and Condition-controlled loops). ● Common arithmetic operators (+, -, *, /, MOD, DIV, ^) and comparison/logical operators. ● The features of fundamental data types (Integer, Real, Boolean, Character, String) and the necessity of explicit type casting. ● String handling techniques (length, substrings, character position, case manipulation). ● The construction and manipulation of 1D and 2D arrays (indexing, iterating, updating records). ● How to read from and write to external text files. ● The purpose and benefits of using modular subprograms (functions and procedures), including the passing of local variables and parameters. ● The purpose of random number generation routines.
Pupils will know how to (procedural knowledge)	● Translate abstract pseudocode or OCR ERL instructions into fully

	functional, clean Python program structures. ● Construct multidimensional array configurations to store and retrieve data ● Write modular functions that accept parameters and return computed data values back to the main program stream.
Core knowledge to know and remember to progress to later learning (foundational knowledge)	● Designing, nesting, and organising logical programming code sequences to handle variable data values predictably at runtime.
Prior knowledge	Y11 Unit 2.1 Algorithms (translating logic charts to code) and KS3 Thread: Computer Science (Textual Python execution foundations covered across Key Stage 3).
Future knowledge	**Linked Units:** Y11 Unit 2.3 Robust Programs (securing inputs) and A Level Unit 1.2 Paradigms (Object-Oriented Programming, classes, objects, instantiation, and method overriding).
Key vocabulary	Sequence, Selection, Iteration, Casting, MOD, DIV, String Handling, 1D Array, 2D Array, File Handling, Subprogram, Function, Procedure, Parameter, Return.

Topic Title	**2.3 Producing Robust Programs**
Pupils will know	● The definitive purpose of defensive design: protecting software systems from malicious inputs and accidental runtime user behaviour. ● Defensive techniques: Input validation (range check, length check, presence check, format check), input sanitisation, and secure user authentication systems. ● The importance of code maintainability features: meaningful variable names, appropriate indentation, comprehensive code commenting, and logical subprogram modularisation. ● The classification of coding errors: Syntax errors (violating language grammar) vs. Logic errors (unexpected code behaviour during successful execution). ● The purpose and structure of comprehensive software testing, including the formulation of test plans. ● The three classifications of testing data inputs: Normal, Boundary, and Erroneous data.
Pupils will know how to (procedural knowledge)	● Embed data validation loops inside programs to trap invalid user inputs before they crash system operations. ● Draft structured test plans containing expected outcomes to identify syntax and logic errors.

Core knowledge to know and remember to progress to later learning (foundational knowledge)	• Designing and thoroughly testing code structures to build resilient software modules that do not fail when presented with unpredictable real-world data inputs.
Prior knowledge	Y10 Unit 1.4 Network Security (understanding malware vectors like SQL Injection) and Y11 Unit 2.2 Programming Fundamentals (core coding execution constructs).
Future knowledge	**Linked Units**: A Level Component 03 NEA (where students must run their own multi-stage development life cycle featuring detailed test plans and validation matrices).
Key vocabulary	Defensive Design, Validation, Sanitisation, Authentication, Maintainability, Syntax Error, Logic Error, Test Plan, Normal Data, Boundary Data, Erroneous Data.

Year 11 overview

Topic Title	**1.3 Computer Networks, Connections and Protocols**
Pupils will know	• The definitions, differences, and physical setups of Local Area Networks (LANs) and Wide Area Networks (WANs). • Factors that affect network performance (bandwidth, number of users etc). • The characteristics of client-server and peer-to-peer network models. • The hardware components required to connect a network: NIC, transmission media, switches, routers, and wireless access points. • The physical and logical layouts of Star and Mesh topologies. • The concept of the internet as a worldwide collection of computer networks, including the role of DNS, hosting, and the cloud. • Core communication protocols:, TCP/IP, HTTP, HTTPS, FTP, POP, IMAP, SMTP and layers. • IP/MAC addressing and the concept of packet switching.
Pupils will know how to (procedural knowledge)	• Identify networks based on their physical setups and characteristics. • Identify the appropriate protocols applied to a data packet as it moves through a network.

Core knowledge to know and remember to progress to later learning (foundational knowledge)	• How physical machines share resources and communicate globally via standardised architectural rules and protocols.
Prior knowledge	Year 8 Hardware & Networking
Future knowledge	**Linked Units**: Y10 Unit 1.4 Network Security (protecting these connection pathways) and A Level Unit 1.3/1.5 Exchanging Data (routing algorithms, DNS lookups, TCP/IP stack layers).
Key vocabulary	LAN, WAN, Client-Server, Peer-to-Peer, Topology, Star, Mesh, Router, Switch, Protocol, IP Address, MAC Address, Packet Switching, DNS.

Topic Title	**1.4 Network Security**
Pupils will know	• Forms of attack and threats to networks: Malware (Viruses, Worms, Trojans, Ransomware, Spyware), Phishing, Brute-Force attacks, Denial of Service (DoS) attacks, Data interception, and SQL Injection. • The human vulnerabilities that lead to security breaches (Social Engineering, poor password hygiene). • Methods to identify and prevent network vulnerabilities: Firewalls, Anti-malware software, User access levels, Passwords, Encryption, and Penetration Testing.
Pupils will know how to (procedural knowledge)	• Analyse real world corporate network setups to flag potential entry vulnerabilities and specify targeted technical defense solutions. • Differentiate between administrative security policies and active technical defense controls.
Core knowledge to know and remember to progress to later learning (foundational knowledge)	• The specific systemic vulnerabilities that exist at the intersection of network hardware, software design, and human interaction.
Prior knowledge	Y10 Unit 1.3 Networks (understanding ports, routers, and data packets) and KS3 Thread: Digital Literacy (Year 7 E-safety rules and password strength guidelines).
Future knowledge	**Linked Units:** Y11 Unit 2.3 Producing Robust Programs (writing defensive code to prevent exploits like SQL Injection) and A Level Unit 1.3 Exchanging Data (advanced encryption standards and hashing algorithms).

Key vocabulary	Malware, Ransomware, Phishing, Brute-Force, DoS, SQL Injection, Social Engineering, Firewall, Anti-malware, Access Levels, Encryption, Penetration Testing.

Topic Title	**1.6 Ethical, Legal, Cultural and Environmental Concerns**
Pupils will know	• How to investigate and discuss computer science technologies while considering wider ethical, cultural, environmental, and privacy impacts. • The scope, purpose, and severe penalties associated with core UK legislation: Data Protection Act 2018, Computer Misuse Act 1990, and Copyright Designs and Patents Act 1988. • The distinct operational and commercial differences between Open-Source and Proprietary software models.
Pupils will know how to (procedural knowledge)	• Synthesise structured, balanced long mark extended writing responses evaluating how modern computing trends impact privacy, society, and environment. • Categorise technological scenarios under their respective, governing legislative acts to flag legal compliance issues.
Core knowledge to know and remember to progress to later learning (foundational knowledge)	• The professional ethical framework and rigid legal boundaries within which every computer scientist and commercial software company must operate.
Prior knowledge	All Year 10 units (understanding the technical systems, security risks, and storage setups that these laws and ethical questions seek to regulate).
Future knowledge	**Linked Units**: A Level Unit 1.5 Legal, Moral, Cultural & Ethical (extended policy frameworks, AI ethics, and international data governance).
Key vocabulary	Ethical, Cultural, Environmental, Privacy, Data Protection Act, Computer Misuse Act, Copyright, Open-Source, Proprietary, Stakeholder.

Topic Title	**2.4 Boolean Logic**
Pupils will know	• The functionality and behaviour of the three fundamental logic gates: NOT, AND, and OR. • The graphical circuit symbols used to represent individual logic gates inside schematic diagrams. • The structure, purpose, and construction rules of multi-variable Truth Tables. • How to combine up to three logic gates together to build composite hardware circuit statements.

Pupils will know how to (procedural knowledge)	• Construct complete, multi-row truth tables to evaluate the output behaviour of layered logic gate arrangements. Interpret complex, written logic expressions and draw their corresponding graphical circuit layouts.
Core knowledge to know and remember to progress to later learning (foundational knowledge)	• Understanding how algebraic and programmatic logic decisions translate directly into physical hardware switching states.
Prior knowledge	Y10 Unit 1.1 Systems Architecture (understanding how transistors switch current inside CPU registers) and Y11 Unit 2.2 Programming Fundamentals (using Boolean conditions in IF statements).
Future knowledge	**Linked Units**: A Level Unit 1.4 Data Types & Structures (Advanced Boolean Algebra simplification theorems, Karnaugh Maps, De Morgan's laws).
Key vocabulary	Logic Gate, NOT, AND, OR, Circuit Diagram, Truth Table, Input, Output, Boolean Expression, Logic State.

Topic Title	**2.5 Languages and Integrated Development Environments**
Pupils will know	• The key functional characteristics of High-Level programming languages vs. Low-Level machine architectures (Machine Code and Assembly Language). • The explicit purpose of language translators in computing systems. • The differences, execution traits, advantages, and disadvantages of Compilers vs. Interpreters. • The core features provided by an Integrated Development Environment (IDE) that assist developers: source code editors, error diagnostics, runtime environments, translators, and debugging facilities (breakpoints, variable watch).
Pupils will know how to (procedural knowledge)	• Utilise advanced IDE tools—such as breakpoints, step-through traces, and variable watch windows—to troubleshoot runtime software bugs. • Choose the correct translation engine (Compiler vs. Interpreter) for a specific deployment scenario, defending the choice based on execution speed and platform independence.
Core knowledge to know and remember to progress to later learning (foundational knowledge)	• Understanding how human-readable source code undergoes conversion into CPU-executable binary instructions.

Prior knowledge	*Y10 Unit 1.1 Systems Architecture* (CPU execution rules) and *Y11 Unit 2.2 Programming Fundamentals* (writing high-level code).
Future knowledge	**Linked Units:** A Level Unit 1.2 Software & Development (compilation sub-phases: Lexical, Syntactic, and Semantic analysis; linkers/loaders) and Unit 1.2 Paradigms (Assembly routines and functional frameworks).
Key vocabulary	High-Level Language, Low-Level Language, Machine Code, Assembly, Translator, Compiler, Interpreter, IDE, Code Editor, Debugging, Breakpoint, Diagnostics.

Topic Title	**Programming projects (throughout the course)**
Pupils will know	• How to decompose a problem into manageable pieces. • The structural conventions of planning multi-routine programs using formal visual flowcharts and structured pseudocode (OCR Exam Reference Language). • How to translate algorithmic designs into syntax-accurate high-level text code (Python). • The necessity of file-handling routines (open, read, write, close) to achieve data persistence across runtime sessions. • The principles of defensive design, including data validation (range, length, presence, format checks), and user authentication. • Code maintainability standards, including variable naming conventions, indentation, and commenting. • The structural differences between Syntax errors, Runtime crashes, and Logic errors. • The boundaries of a comprehensive testing matrix, classifying input vectors into Normal, Boundary, Invalid and Erroneous data profiles. • The difference between iterative testing and terminal testing.
Pupils will know how to (procedural knowledge)	• Break a large programming problem down into manageable, self-contained subprograms (functions and procedures) that pass parameters and return values. • Construct and manipulate 1D and 2D arrays to store, retrieve, and update tabular records programmatically. • Write validation routines that handle invalid user input without throwing runtime crashes or index errors. • Populate a formal, multi-column test matrix with precise input values, document actual vs. expected outcomes, and use IDE debugging tools (breakpoints, variable watch streams) to isolate and refactor logic bugs.
Core knowledge to know and remember to progress	• Software engineering requires a blueprint before coding, which must then be systematically validated using defensive programming structures and an objective testing plan.

to later learning (foundational knowledge)	
Prior knowledge	**Linked Units:** Component 02 Written Examination Section B (Answering high-tariff 6-mark and 20-mark code completion, debugging, and logic design questions) and A Level Component 03 NEA (The independent software engineering lifecycle project).
Future knowledge	**Linked Units:** *A Level Component 03 NEA* (Formally documenting a comprehensive multi-tier user analysis and system success criteria).
Key vocabulary	Analysis, Decomposition, Abstraction, Pseudocode, OCR ERL, Subprogram, Parameter, Return Value, 2D Array, Data Persistence, File I/O, Input Validation, Maintainability, Refactoring, Test Matrix, Normal Data, Boundary Data, Erroneous Data, Logic Error, Debugging.