Programme of study for **Year 11 Computer Science**

OCR GCSE (9–1) Computer Science J277 – Component 2: Computational thinking, algorithms and programming (J277/02)

Autumn (1st term) Topic	Autumn (2nd term) Topic	Spring (1st term) Topic	Spring (2nd Term) Topic	Summer (1st term) Topic	Summer (2nd term) Topic
Component 2: **2.1.1 Computational thinking** Abstraction, decomposition, algorithmic thinking **2.1.2 Designing, creating and refining algorithms** Inputs, processes and outputs; structure diagrams; flowcharts; pseudocode and OCR Exam Reference Language; trace tables; identifying and correcting errors **2.2.1 Programming fundamentals** Variables, constants, inputs, outputs, assignment; sequence, selection, iteration (count- and condition-controlled, nesting); arithmetic, comparison and Boolean operators **2.2.2 Data types** Integer, real, Boolean, character, string; casting	**Component 2:** **2.2.3 Additional programming techniques (1)** String manipulation (concatenation, slicing); file handling (open, read, write, close); 1D and 2D arrays; records; SQL (SELECT, FROM, WHERE); random number generation **2.4.1 Boolean logic** AND, OR, NOT logic diagrams and gate symbols; truth tables; combining operators	**Component 2:** **2.2.3 Additional programming techniques (2)** Sub programs: functions and procedures; local and global variables and constants; passing and returning arrays **2.1.3 Searching and sorting algorithms** Linear and binary search; bubble, merge and insertion sort **2.3.1 Defensive design** Anticipating misuse; input validation; authentication; maintainability (sub programs, naming conventions, indentation, commenting)	**Component 2:** **2.3.2 Testing** Purpose of testing; iterative and final testing; syntax and logic errors; normal, boundary, invalid and erroneous test data; test plans; refining algorithms **2.5.1 Languages** High- and low-level languages; translators; compiler and interpreter **2.5.2 The Integrated Development Environment (IDE)** Editors, error diagnostics, run-time environment, translators **Practical Programming project** Design, write, test and refine a program to a specification or to solve a problem, in a high-level text-based language (Python)	**Components 1 and 2:** **Revision and exam preparation** J277/01: 1.1–1.6 (taught in Year 10) J277/02: 2.1–2.5 **Exam technique** Command words; calculation and scenario questions; Section B programming and algorithm questions under timed conditions	**GCSE examinations:** **GCSE examination series** J277/01 Computer systems (1 hour 30 minutes, 80 marks) J277/02 Computational thinking, algorithms and programming (1 hour 30 minutes, 80 marks) **Final revision** Lessons, where timetabled, are used for exam-technique and last-minute retrieval sessions
End of term 1 evidence to cover: Mock assessment: J277/02-style questions on 2.1.1, 2.1.2, 2.2 (including 2.2.3 techniques taught so far) and 2.4.1 Boolean logic, plus retrieval of Year 10 Component 1 content. Practical coding evidence from Python tasks.		**End of term 2 evidence to cover:** Full J277/02-style paper (80 marks, 1 hour 30 minutes, Sections A and B, non-calculator) covering 2.1–2.5. Practical Programming project and evidence for the centre's Practical Programming Statement.		**End of year evidence to cover:** GCSE examinations: J277/01 and J277/02. Prior to this: full mock papers for both components, with question-level analysis used to target revision.	

Rationale for sequence:	**Rationale for sequence:**	**Rationale for sequence:**	**Rationale for sequence:**	**Rationale for sequence:**	**Rationale for sequence:**
Opens with the thinking and design skills every other topic relies on, and the ways algorithms are written (flowcharts, pseudocode, OCR Exam Reference Language), which is also how Paper 2 questions are set. Practical Python coding follows straight away, building on Year 9, so students apply design to working programs. Data types come before strings, arrays and files. Trace tables are introduced on simple algorithms, then reused once loops and selection are secure.	Extends the core constructs to storing and processing collections of data: strings, then files, then arrays, then 2D arrays and records, then SQL (2D arrays emulate database tables, which SQL then queries), then random numbers. Arrays need to be secure before searching and sorting in the next half-term. Boolean logic is a short "unplugged" block that revisits AND, OR and NOT from the first half-term as diagrams and truth tables, and gives a change of pace before the mock.	Sub programs come first so searching and sorting can be written as reusable functions that work on the arrays from the previous half-term. This keeps the Year 9 principle of coding before algorithms, so students implement standard algorithms rather than only describing them. Defensive design follows naturally: validation and authentication are written as sub programs, and maintainability applies to everything written so far.	Testing follows defensive design, as students test the validation they have built. IDE error diagnostics support debugging, which links to syntax and logic errors, and compilers and interpreters explain how Python code runs. The Practical Programming project then brings all the content together through design, write, test and refine, the skills assessed in Section B of Paper 2.	All content is taught. Revision is planned from mock data, with Component 1 retrieval (taught in Year 10) alongside Component 2 so both papers are covered before the exams.	Examination series.
Home – Learning: Python practice tasks on fundamentals and data types; flowchart and pseudocode questions in OCR Exam Reference Language. Component 1 retrieval: 1.1 Systems architecture and 1.2.1–1.2.2 Memory and storage.	Home – Learning: Programs using arrays, files and strings; SQL query questions; truth table and logic diagram practice. Component 1 retrieval: 1.2.3–1.2.5 Data representation and compression.	Home – Learning: Write and trace search and sort algorithms; validation and authentication coding tasks. Component 1 retrieval: 1.3 Networks and 1.4 Network security.	Home – Learning: Test plans and test data for a given scenario; project work at home as directed. Component 1 retrieval: 1.5 Systems software and 1.6 Impacts of digital technology.	Home – Learning: Past-paper questions from both papers, selected from mock analysis; revision plan.	Home – Learning: Revision as directed.

Reading / High Quality Text: Specification J277, section 2c: 2.1.1–2.2.2. Specification J277, section 3c: OCR Exam Reference Language.	Reading / High Quality Text: Specification J277, section 2c: 2.2.3 and 2.4.1.	Reading / High Quality Text: Specification J277, section 2c: 2.1.3, 2.2.3 and 2.3.1.	Reading / High Quality Text: Specification J277, section 2c: 2.3.2 and 2.5; sections 2d and 3b: Practical Programming skills.	Reading / High Quality Text: Past papers, mark schemes and examiners' reports on the OCR website.	Reading / High Quality Text: Past papers and mark schemes on the OCR website.
Numeracy: Arithmetic operators including MOD, DIV and exponentiation; integer v real; order of operations.	Numeracy: Truth tables (0 and 1); 2D array row and column positions; ranges for random numbers.	Numeracy: Binary search: finding the midpoint and halving the list; counting passes and comparisons in sorts.	Numeracy: Boundary values and ranges when choosing test data.	Numeracy: Non-calculator Paper 2 practice: trace tables, MOD and DIV, Paper 1 conversions and file sizes.	Numeracy: Exam period.
Enrichment / opportunities to develop cultural capital (including careers, WRL and SMSC): Careers (nationalcareers.service.gov.uk job profiles): software developer, games programmer, database administrator, data analyst. Post-16 pathways: A Level Computer Science and BTEC IT taster sessions, and options guidance, in the Autumn and Spring terms. WRL: link programming and project work to local employers and apprenticeships. SMSC: ethical use of data in programs and databases.					