Key Stage 4 Computing Skill Matrix (GCSE)Key Stage 4 Computing Skill Matrix (GCSE)

	Emerging	Developing	Secure	Mastered
Algorithms	Identify and apply algorithms to solve simple computational problems. Analyse simple algorithms by identifying their inputs, processing steps, and outputs, and represent them using flowcharts or pseudocode.	Break down complex problems into smaller, manageable parts using decomposition, and design solutions for each part. Demonstrate how to perform linear and binary search, in a practical physical manner and use it to find values in a dataset, explaining how the algorithm processes the data.	Identify and remove unnecessary detail from a problem to create a simplified model and use this abstraction to design a solution. Demonstrate how to perform merge and bubble sort, in a practical physical manner and use it to find values in a dataset, explaining how the algorithm processes the data.	Use a systematic approach to problem solving and algorithm creation representing those algorithms using pseudo-code and flowcharts. Compare multiple algorithms that solve the same problem and justify which is more appropriate based on context (e.g., efficiency, simplicity).
Computer Systems	Describe the role of the CPU and its components (ALU, control unit, clock, registers, buses), explain the Fetch-Execute cycle, and evaluate how factors like clock speed, cores, and cache size affect performance. Identify and explain the purpose of different types of memory (RAM, ROM, cache, registers) and distinguish between main memory and secondary storage. Compare types of secondary storage (solid state, optical, magnetic) in terms of operation, advantages, and disadvantages.	Identify when interpreters, compilers, and assemblers are used and describe how code is translated for execution. Distinguish between system software and application software, giving examples of each, and explain the role of operating systems and utility programs in managing hardware, software, and system security.	Compare low-level and high-level programming languages, including machine code and assembly language, and explain the advantages and disadvantages of each. Define computer networks and describe the main types (PAN, LAN, WAN). Compare wired and wireless networks, including their advantages and disadvantages.	Explain the purpose of common network protocols (e.g., Ethernet, Wi-Fi, TCP, IP, HTTP/S) and describe how they operate within the four-layer TCP/IP model. Explain the importance of network security and describe methods such as authentication, encryption, firewalls, and MAC address filtering. Explain the concept of cloud storage and evaluate its benefits and limitations compared to local storage.
Databases	Explain the purpose of databases and relational databases, using key concepts such as tables, records, fields, primary keys, and foreign keys.	Describe how relational databases reduce data redundancy and inconsistency.	Use SQL to manage data in a relational database, including: • Retrieving data using SELECT, FROM, WHERE, and ORDER BY clauses • Inserting data using INSERT INTO and VALUES • Updating and deleting data using UPDATE, SET, and DELETE FROM with conditions	Create, design and evaluate tables within a relational database, using key concepts correctly, to reduce data redundancy and inconsistency.

Key Stage 4 Computing Skill Matrix (GCSE)Key Stage 4 Computing Skill Matrix (GCSE)

Data Representation	Convert between decimal, binary, and hexadecimal number systems, and explain how and why binary and hexadecimal are used to represent data and instructions in computing. Perform binary addition and apply binary shifts to binary numbers, explaining how shifts affect the value and how they are used in computing contexts.	Compare and calculate data sizes using bits, bytes, and standard prefixes (kilo, mega, giga, tera), and explain how these units are used to measure digital information.	Describe how sound is digitally represented using sampling rate and sample resolution and calculate sound file sizes based on these properties. Represent bitmap images using pixels and colour depth, calculate image file sizes based on resolution and colour depth, and convert between binary data and bitmap representations.	Apply Huffman coding and run length encoding (RLE) to compress data, interpret Huffman trees, and calculate the number of bits required to store compressed and uncompressed data.
Ethics and Security	Define cyber security and explain its main purposes, including protecting systems, networks, and data from digital threats.	Describe common types of malware (e.g., viruses, trojans, spyware), explain how they operate, and suggest protective measures. Identify a range of cyber security threats including pharming, weak passwords, misconfigured access rights, removable media, and outdated software.	Identify and explain different forms of social engineering (e.g., blagging, phishing, shoulder surfing), describe how they exploit human behaviour, and suggest ways to protect against them. Explain the purpose of penetration testing and describe how it is used to identify vulnerabilities.	Evaluate the ethical, legal, and environmental impacts of digital technology on individuals and society, including issues related to data privacy, security, and sustainability. Evaluate the effectiveness of security measures such as biometrics, password systems, CAPTCHA, email confirmations, and automatic software updates.
Programming	Declare and manipulate variables and constants using appropriate data types (integer, real, Boolean, character, string), apply arithmetic operators to perform calculations, and use meaningful identifiers to improve code readability and maintainability. Write programs that interact with users through input and output and identify and correct syntax and logic errors through testing and debugging.	Write programs that use selection (with comparison and logical operators) and iteration (definite and indefinite) to control program flow and solve problems. Identify and correct syntax and logic errors in code through testing and debugging with support.	Use string handling techniques in programs, including measuring length, finding character positions, extracting substrings, concatenating strings, converting characters to and from character codes, and performing string-to-number and number-to-string conversions. Use arrays (or equivalent data structures) to store and manipulate collections of data and incorporate random number generation in programs to introduce variability or simulate real-world scenarios.	Design and implement structured programs using subroutines that include parameters and return values. Use local variables within subroutines and explain how modular design improves readability, reusability, and maintainability.