KS4 Module 1: Programming techniques *(Year 10 Module 1 of 4)*	KS4 Module 2: Computational Thinking *(Year 10 Module 2 of 4)*	KS4 Module 3: Data *(Year 10 Module 3 of 4)*
Knowledge *What pupils will know*	**Knowledge *What pupils will know***	**Knowledge *What pupils will know***
• How to analyse and apply abstraction to a problem scenario • How to analyse apply decomposition to a problem solution • How to solve problems • How to convert algorithms into programs • How to make programs maintainable • How to identify, locate, and correct program errors • Apply logical reasoning to test data to evaluate a programs fitness for purpose • The difference between, and use of, both global & Local variables	• The benefits of using decomposition and abstraction to model aspects of the real world • The benefits of using subprograms • How to follow and write algorithms in both pseudocode and flowchart formats • The need for and benefits of using variables, constants, 1D and 2D lists, and records. • The need for and common applications of arithmetic, relational, and logical operators • How to use trace tables and develop effective test data to determine the outputs for given algorithms and identify errors in code to determine suitability for purpose • How Bubble sort, Merge sort, Linear search & Binary search work	• Why computers use binary, and its limitations • How the binary number system is used to represent values, including the maximum number of states that can be represented by a binary pattern of a given length • How computers represent positive and negative numbers in binary • The concept of overflow in relation to bit storage and the impact of this concept on arithmetic operations • Why hexadecimal notation is used in computer systems, including common applications of hex. • The need for data compression and it's impact on storage size, with reference to measures in binary multiples
Skill *What pupils will be able to do*	**Skill *What pupils will be able to do***	**Skill *What pupils will be able to do***
• Read, write, analyse and refine programs in Python • Write programs that make effective use of sequence, selection, and iteration techniques • Write programs that make effective use of subroutines • Write programs that make appropriate use of variables & constants, including correct use of data types and casting. • Write programs that manipulate strings • Write programs that read from and write to external files, including .csv files • Implement validation and authentication techniques • Effectively use logical operators in programs • Write programs that make appropriate use of lists (both 1D & 2D)	• Apply logical operators in truth tables with up to three inputs • Write simple searching and sorting algorithms • Use trace tables • Resolve syntax, logic, and runtime errors in code • Use an IDE to generate efficient programs • Write programs to solve a given problem • Determine the most suitable levels of abstraction to apply to a given problem • Decompose a problem into appropriate modular solutions	• Convert numbers between different bases (Denary, Binary, Hexadecimal) • Represent both positive and negative numbers in Binary, using both sign and magnitude and two's complement • Represent images and sounds as a series of binary digits • Re-create images and sounds from a series of binary digits • Convert text to binary and vice versa using 7-bit ASCII • Perform arithmetic operations on numbers in binary • Apply simple lossless compression techniques to data files

KS4 Module 4: Computers *(Year 10 Module 4 of 4)*	KS4 Module 5: Networks *(Year 11 Module 1 of 2)*	KS4 Module 6: Issues & Impact *(Year 11 Module 2 of 2)*
Knowledge ***What pupils will know***	**Knowledge** ***What pupils will know***	**Knowledge** ***What pupils will know***
• The stored program and the role of main memory • The components (Units, Registers, & Buses) that make up the Von-Neumann CPU, and how they interact to complete the Fetch-Decode-Execute cycle. • The role of secondary storage and the way data is stored on each category of storage device • The concept of an embedded system • The purpose and function of the operating system • The purpose and function of utility software • The importance of developing robust software and methods of identifying vulnerabilities • The characteristics and purposes of both low-level and high-level programming languages • The differences between interpreters and compilers in the way it translates source code to object code	• Why computers are connected in networks • The different types of networks, such as LAN's and WAN's • How the internet is structured, including the use of IP & MAC addresses, routers, and the Domain Name System • The characteristics of wired and wireless networks, hand the impact of connection medium on performance • How to measure and calculate network transmission speeds • The role of and need for network protocols, in particular the 4 layer TCP/IP model • The characteristics of network topologies, including bus, mesh and star topologies) • The importance of network security • Ways of identifying network vulnerabilities • Methods of protecting networks	• Environmental issues associated with the use of digital devices • Ethical and legal issues associated with the collection and use of personal data • Methods of intellectual property protection for computer systems and software • The threats to digital systems posed by malware • How hackers exploit vulnerabilities in systems • How social engineering is used to carry out cyber attacks • Methods of protecting digital systems and data.
Skill ***What pupils will be able to do***	**Skill** ***What pupils will be able to do***	**Skill** ***What pupils will be able to do***
• Write simple programs in assembly language using the LMC CPU simulation • Model the FDE cycle in a Von-Neumann CPU • Perform translations on simple blocks of source code to object code	• Connect devices together to form basic networks • Model network transmissions using microcontrollers • Demonstrate the application of packet switching and circuit switching • Construct and manipulate expressions involving file size, transmission rate, and time	• Conduct small scale investigations into system vulnerabilities and attacks • Create simple security/risk status reports for given systems • Develop tools to support system security, such as simple authentication or 2fA programs. • Apply copyright/licencing processes to digital works through creative commons licencing methods.