Computer Science Curriculum Intent

Technology is changing the lives of everyone. Through computing we aim to equip students to participate in a rapidly changing world where work and leisure activities are increasingly transformed by technology. The key aims for GCSE Computing are to provide a high-quality computing education which equips students to use computational thinking and creativity to understand and change the world. The curriculum will teach students key knowledge about how computers and computer systems work, and how they are designed and programmed.

Relevant to the modern, changing world of computing, the qualification is designed to boost computing skills essential for the modern world. Computing companies, organisations, academics, and teachers have been involved in shaping and developing this contemporary qualification. The GCSE in Computer Science focuses on computational thinking as its core, helping students to develop the programming skills to solve problems, design systems and understand human and machine intelligence. A breakdown of the units studied in years 10 and 11 is given below.

Through this curriculum we aim to give our students the life-skills that will enable them to embrace and utilise new technology in a socially responsible and safe way in order to flourish. We want our students to be able to operate in the 21st century workplace and we want them to know the career opportunities that will be open to them if they study computing. Not only do we want them to be digitally literate and competent end-users of technology but through our computer science lessons we want them to develop creativity, resilience and problem-solving and critical thinking skills.

Year 10	Half Term 1	Half Term 2	Half Term 3	Half Term 4	Half Term 5	Half Term 6
Knowledge Introduced	2.4.1- Boolean logic 1.2.3 – Units 1.2.4 – Data Storage – Numbers 1.2.4 – Data Storage – Characters 2.1.1 – Computational Thinking **Practical Programming Skills**	1.2.4 - Data storage – Images 1.2.4 - Data storage – Sound 1.2.5 - Data storage – Compression 2.1.2 Designing, creating and refining algorithms.	1.1.1- Architecture of the CPU 1.1.2- CPU Performance 2.2.1 -Programming fundamentals 2.2.2 -Data types	1.1.3- Embedded systems 1.2.1- Primary storage (Memory) 1.2.2- Secondary storage 2.2.1- Programming fundamentals 2.2.3- Additional programming techniques	1.2.2- Secondary storage 1.3.1- Networks and topologies 2.2.3 -Additional programming techniques **Practical Programming Skills**	1.3.1- Networks and topologies 1.3.2 -Wired and wireless networks, protocols and layers Exam Prep Year 10 Mock
Key vocabulary/ concepts/ideas students must master	Knowledge of the truth tables for each logic gate. Recognition of each gate symbol Simple logic diagrams using the operators AND, OR and NOT. Truth tables. Combining Boolean operators using AND, OR and NOT. Applying logical operators in truth tables to solve problems. Understanding of how to create, complete or edit logic diagrams and truth tables for given scenarios. Ability to work with more than one gate in a logic diagram. The units of data storage (Why data must be stored in binary format): Bit, Nibble (4 bits), Byte (8 bits), Kilobyte (1,000 bytes or 1 KB), Megabyte (1,000 KB), Gigabyte (1,024 MB), Terabyte (1,024 GB), Petabyte (1,024 TB) The use of binary codes to represent characters. The term 'character set'. The relationship between the number of bits per character in a character set, and the number of characters which can be represented, e.g. ASCII, Unicode How to convert positive denary whole numbers to binary numbers (up to and including 8 bits) and vice versa. How to add two binary integers together (up to and including 8 bits) and explain overflow errors which may occur. How to convert positive denary whole numbers into 2-digit hexadecimal numbers and vice versa. How to convert binary integers to their	Images - How an image is represented as a series of pixels, represented in binary. Metadata. The effect of colour depth and resolution on: The quality of the image, The size of an image file Sound - How sound can be sampled and stored in digital form. The effect of sample rate, duration, and bit depth on: The playback quality, The size of a sound file Compression - The need for compression. Types of compression: Lossy, Lossless Designing, creating, and refining algorithms - Identify the inputs, processes, and outputs for a problem. Structure diagrams. Create, interpret, correct, complete, and refine algorithms using: Pseudocode, Flowcharts, Reference language/high-level programming language. Identify common errors. Trace tables.	Architecture of the CPU - The purpose of the CPU: The fetch-execute cycle (What actions occur at each stage of the fetch-execute cycle) Common CPU components and their function: ALU (Arithmetic Logic Unit), CU (Control Unit), Cache, Registers (The role/purpose of each component and what it manages, stores, or controls during the fetch-execute cycle) Von Neumann architecture: MAR (Memory Address Register), MDR (Memory Data Register), Program Counter, Accumulator (The purpose of each register, what it stores (data or address), The difference between storing data and an address) CPU Performance - How common characteristics of CPUs affect their performance: clock speed, cache size, number of cores (Understanding of each characteristic as listed) (The effects of changing any of the common characteristics on system performance, either individually or in combination Programming fundamentals - The use of variables, constants, operators, inputs, outputs and Assignments. The use of the three basic programming constructs used to control the flow of a program: Sequence, Selection, Iteration (count- and condition-controlled loops). The common arithmetic operators. The	Embedded Systems - Give examples of embedded systems (Familiarity with a range of different embedded systems) Primary Storage - The need for primary storage (Why computers have primary storage, How this usually consists of RAM and ROM), The difference between RAM and ROM (Key characteristics of RAM and ROM), The purpose of ROM in a computer system, The purpose of RAM in a computer system, Virtual memory (Why virtual memory may be needed in a system: How virtual memory works, Transfer of data between RAM and HDD when RAM is filled) Secondary Storage - The need for secondary storage (Why computers have secondary storage), Common types of storage: Optical, Magnetic, Solid state (Recognise a range of secondary storage devices/media, Differences between each type of storage device/medium). Additional techniques - The use of basic string manipulation. The use of basic file handling operations: Open, Read, Write, Close. The use of records to store data. The use of SQL to search for data. The use of arrays (or equivalent) when solving problems, including both one-dimensional and two-dimensional arrays. How to use sub programs (functions and procedures) to produce structured code. Random number generation	Secondary Storage - Compare advantages/disadvantages for each storage device, be able to apply their knowledge in context within scenarios. Networks and Topologies - Types of networks: LAN (Local Area Network), WAN (Wide Area Network). Factors that affect the performance of networks. The different roles of computers in a client-server and a peer-to-peer network. The hardware needed to connect stand-alone computers into a Local Area Network: Wireless access points, Routers, Switches, NIC (Network Interface Controller/Card), Transmission media. The Internet as a worldwide collection of computer networks: DNS (Domain Name Server), Hosting, The Cloud, Web servers and clients. Star and Mesh network topologies. Additional techniques - The use of basic string manipulation. The use of basic file handling operations: Open, Read, Write, Close. The use of records to store data. The use of SQL to search for data. The use of arrays (or equivalent) when solving problems, including both one-dimensional and two-dimensional arrays. How to use sub programs (functions and procedures) to produce structured code. Random number generation	Wired and wireless Networks, protocols, and layers - Modes of connection: Wired, Ethernet, Wireless. Wi-Fi, Bluetooth. Encryption. IP addressing and MAC addressing. Standards. Common protocols including: - TCP/IP (Transmission Control Protocol/Internet Protocol), HTTP (Hyper Text Transfer Protocol), HTTPS (Hyper Text Transfer Protocol Secure), FTP (File Transfer Protocol), POP (Post Office Protocol), IMAP (Internet Message Access Protocol), SMTP (Simple Mail Transfer Protocol), The concept of layers.

	hexadecimal equivalents and vice versa. Binary shifts Principles of computational thinking: Abstraction, Decomposition, Algorithmic thinking		common Boolean operators AND, OR and NOT. Data Types - The use of data types: Integer, Real, Boolean, Character and string, Casting			
Knowledge synoptic links	Architecture of the CPU and Memory Practical programming	Architecture of the CPU and Memory Practical programming	Data storage and Memory Practical programming	Practical programming Data storage and Memory	Cyber security Practical programming	Cyber security
CEIAG Links/ Opportunities	GB4 – Software Developer GB4 - Hardware designer GB4 – Web developer	GB4 - Software Developer GB4 – Graphics designer	GB4 - Software Developer GB4 - Hardware designer	GB4 - Software Developer GB4 – Systems designer GB4 – App Lab	GB4 - Software Developer GB4 – E-sports Gamer	GB4 - Network Engineer GB4 – Cyber security GB4 – Ethical hacker GB4 – Information security analyst

Year 11	**Half Term 1**	**Half Term 2**	**Half Term 3**	**Half Term 4**	**Half Term 5**	**Half Term 6**
Knowledge Introduced	1.4.1 - Threats to computer systems and networks 1.4.2 - Identifying and preventing vulnerabilities 1.5.1- Operating systems 2.3.1- Defensive design 2.3.2- Testing	1.5.2- Utility software 1.6.1- Ethical, legal, cultural and environmental impact 2.3.2- Testing 2.5.1- Languages 2.5.2 -The Integrated Development Environment (IDE)	Programming Revision 2.1.3 -Searching and sorting algorithms **Searching and Sorting Practical Programming skills**	Mocks Theory Revision **Practical Programming Skills Revision**	Theory Revision **Practical Programming Skills Revision**	Study leave / exams/collapsed timetable
Key vocabulary/ concepts/ideas students must master	Threats to computer systems and networks Forms of attack: Malware, Social engineering, e.g. phishing, people as the 'weak point', Brute-force attacks, Denial of service attacks, Data interception and theft, The concept of SQL injection Identifying and preventing vulnerabilities Common prevention methods: Penetration testing, Anti-malware software, Firewalls, User access levels, Passwords, Encryption, Physical security	Utility Software - The purpose and functionality of utility software. Utility system software: Encryption software, Defragmentation, Data compression Impacts of digital technology on wider society including Ethical issues, Legal issues, Cultural issues, Environmental issues, Privacy issues. Legislation relevant to Computer Science: The Data Protection Act 2018, Computer Misuse Act 1990, Copyright Designs and Patents Act 1988, Software licences (i.e. open source and proprietary)	Searching and sorting algorithms - Standard searching algorithms: Binary search, Linear search. Standard sorting algorithms: Bubble sort, Merge sort, Insertion sort.			

	Operating Systems - The purpose and functionality of operating systems: User interface, Memory management and multitasking, Peripheral management and drivers, User management, File management Defensive design - Defensive design considerations: Anticipating misuse, Authentication. Input validation. Maintainability: Use of sub programs, Naming conventions, Indentation, Commenting Testing - Identify syntax and logic errors. Selecting and using suitable test data: Normal, Boundary, Invalid, Erroneous, Refining algorithms	Languages - Characteristics and purpose of different levels of programming language: High-level languages, Low-level languages. The purpose of translators. The characteristics of a compiler and an interpreter. The IDE - Common tools and facilities available in an Integrated Development Environment (IDE): Editors, Error diagnostics, Run-time environment, Translators				
Knowledge synoptic links	Practical programming	Practical programming				
CEIAG Links/ Opportunities	GB4 - Cyber Security GB4 – Software Developer GB4 – IT Technician	GB4 – Software Developer GB4 – UX designer GB4 – Cyber Security	GB4 – Software Developer GB4 - Data Scientist/Analysis GB4 – Database Administrator			