Wallington
County Grammar School

The Computer Science Curriculum

Whole School Curriculum Intent:	Wallington County Grammar School is a highly academic but pastorally minded school which delivers a curriculum that enables all students to embody our motto - *Per Ardua ad Summa*, Through Difficulties to the Heights. Each Subject Leader has autonomy over their own curriculum and its intent, i.e. its subject content, skills content, sequencing and assessment schedule. This is vital to ensure the academic curriculum is designed by highly qualified subject experts. The intentions behind whole school approach to curriculum design taken by senior leaders are to provide: ● **Breadth** - We intend to provide a broad, academic and liberal curriculum that equips students with the body of human knowledge and different ways of thinking necessary to succeed in and enjoy their education, careers and wider lives. ● **Depth** - We do not want our students to simply study the national curriculum and examination specifications with grades being our sole focus. We aim for our students to become true scholars of the disciplines that they are learning so that they achieve a deep and sophisticated level of knowledge and understanding. ● **Values** - We aim for our students to develop our four core values: commitment, courage, compassion and creativity. ● **Democracy** - We aim for all our students to have the necessary knowledge and confidence, not just to participate in the democracy of the United Kingdom, but to lead it.
Subject Curriculum Intent:	The intent of our Computer Science curriculum is to inspire **computational thinking**, foster **creativity**, and develop the **critical skills** necessary to understand and shape the digital world. We aim to deliver a curriculum that enables all students to embody our motto - *Per Ardua ad Summa*, Through Difficulties to the Heights. We believe that all students, regardless of background, should be equipped with the fundamental principles of information and computation. By offering a comprehensive Computer Science curriculum, we ensure students have the same opportunities to engage with and contribute to the rapidly evolving technological landscape, promoting a broad and equitable access to a subject vital for modern life. Through the study of computer science principles, practical programming, and the analytical application of information technology, students explore the foundations of the digital world. This includes understanding how digital systems work, the logic behind algorithms, and the profound impact of technology on both natural and artificial systems. Computer Science helps students understand how these foundations continue to shape our world today and will drive its future, preparing them not just to participate in society, but to lead it. Our curriculum enables students to, aligning with our core values of commitment, courage, compassion, and creativity: ● **Think deeply** about information, logic, and systems, fostering a problem-solving mindset through the application of abstraction, logic, algorithms, and data representation. ● **Develop strong analytical and programming skills** through repeated practical experience of writing computer programs to solve a variety of

	computational problems. ● **Make meaningful connections** between theoretical computer science and its real-world applications, understanding its societal implications and ethical considerations. ● **Build confidence** in evaluating and applying new or unfamiliar technologies, developing digital literacy, and acting as responsible, competent, and creative users of information and communication technology. Studying Computer Science nurtures well-rounded, innovative individuals and supports academic success, building on its deep links with mathematics, science, and design and technology. It is also highly respected by top universities, particularly for competitive courses in engineering, technology, and related disciplines, providing a platform for more advanced studies or professional careers. Above all, we want our students to enjoy exploring the power of computational thinking, the logic of algorithms, and the extraordinary potential of technology to innovate and solve problems. We want them to leave with a lasting appreciation of its relevance and its capacity to transform and improve the world, becoming true scholars of the discipline.
Subject Curriculum Aims:	Our Computer Science curriculum is designed to give students a rich and meaningful understanding of the fundamental principles of information and computation. Through the study of computer science principles, practical programming, and the analytical application of information technology, students will explore the foundations of the digital world and develop the skills to understand and shape it, aligning with our motto *Per Ardua ad Summa*, Through Difficulties to the Heights. By the end of their studies, students will be able to: ● **Develop a deep and wide-ranging knowledge** of the fundamental principles and concepts of computer science, including **abstraction, decomposition, logic, algorithms, and data representation**. This involves understanding how digital systems work and how to put this knowledge to use through programming. ● **Analyse problems in computational terms** and apply their understanding through **repeated practical experience of designing, writing, and debugging computer programs** to solve a variety of computational problems. This process cultivates a capacity for **creative, innovative, analytical, logical, and critical thinking**. ● **Understand the components that make up digital systems**, including both hardware and software, and how these components communicate with one another and with other systems. Students will also grasp how various types of data (such as text, sounds, and pictures) can be represented and manipulated digitally in the form of binary digits. ● **Evaluate and apply information technology**, including new or unfamiliar technologies, analytically to solve problems. This empowers them to create programs, systems, and a diverse range of content. ● **Articulate and critically engage with the individual (moral), social (ethical), legal, and cultural opportunities and risks** of digital technology. They will learn a range of ways to use technology safely, respectfully, responsibly, and securely, including protecting their online identity and privacy, and recognising inappropriate content or conduct. ● **Apply essential mathematical skills** relevant to Computer Science, which are embedded throughout the curriculum. This includes the mathematical skills used to express computational laws and processes, such as Boolean algebra and the comparison of the complexity of algorithms. ● **Build strong foundations for further study** and professional careers, gaining the skills and understanding needed to progress to higher levels of study or a professional career in Computer Science and related disciplines. This includes developing their digital literacy at a level suitable for the future workplace and as active participants in a digital world. Computer Science not only helps students appreciate the power and limits of human and machine intelligence and the extraordinary potential of

	technology but also develops critical thinking, problem-solving, and analytical skills that are valuable in many areas of life, learning, and future careers in an increasingly technological society.
Exam Boards	GCSE: OCR A Level: OCR

	Autumn 1	Autumn 2	Spring 1	Spring 2	Summer 1	Summer 2
Y7	**Flowgorithms** Core Programming Constructs Programming Skills in Flowgorithms Flowcharts	**Bebras and Cyber Security** An understanding of cyber security Analysing the vulnerability of a computer system and what can be put in place to help	**Micro-bits** Core programming constructs Programming skills in Java	**Word Processor** Word Processor Functions Skills Explain Target Audience Display Data in a Word Processor Research Data and information to present on a professional document	Code.org Core Programming Constructs Programming skills in Scratch, HTML and Python How to apply Logic, Computability and Algorithms	**Spreadsheets** Functional Spreadsheet skills Spreadsheet Formula and Maths
	Assessment 1 Format:	Multiple choice questions on the course content.		Assessment 2 Format:	Multiple choice questions on the course content.	

	Autumn 1	Autumn 2	Spring 1	Spring 2	Summer 1	Summer 2
Y8	**Python Programming** Core Programing Constructs Programming skills in Python Searching Algorithms and how Binary Search works on a list of numbers		**Boolean and Binary - Understanding Computers** Distinguish between hardware and software, giving examples of computer hardware and software Draw a block diagram showing CPU, input, output and storage devices Name different types of permanent storage device and suggest appropriate input and output devices for a simple scenario Explain what RAM and ROM are used for Show how numbers and text can be represented in binary Explain the impact of future technologies	**AI Prompt Craft** Understand the origin and uses of AI Understand how rules are used in AI decision making Understand what ethics is Consider some simple ethical hypothetical problems Understand how intelligence can be measured in humans and computers Know what the Turing test is and how it works Use Generative AI to solve real world problems		**HTML and Web Development** The core programming constructs Programming skills in HTML
	Assessment 1 Format:	Multiple choice questions and short response questions on the course content		Assessment 2 Format:	Multiple choice questions and short response questions on the course content	
Y9	**Systems Architecture** CPU Operation and Performance Memory and Storage Hierarchy System Applications and Evaluation	**Data Representation** Core Data Representation Encoding Media in Binary Data Management and Integrity	**Algorithms and Problem Solving** Algorithmic Principles Common Searching and Sorting Methods Defining and Tracing Algorithms	**Advanced Level Python** Core Program Logic and Control Flow Data Organization and Handling Modular Code and Best Practices	**T.I.M.E 2 Challenge** How to apply the 5 key principles section to solve complex problems and justifying their approach Using the T.I.M.E approach to programming a solution to a problem Understand how the T.I.M.E (Similar to PRIMM) can be applied in learning a programming language	

	Autumn 1	Autumn 2	Spring 1	Spring 2	Summer 1	Summer 2
	Assessment 1 Format:	Short to middle length response questions adapted from OCR specifications.		Assessment 2 Format:	Short to middle length response questions adapted from OCR specifications.	
Y10	**Networks, connections and protocols** Explain the advantages of networking stand-alone computers into a local area network Explain the difference between a client-server and a peer-to-peer network Describe, using diagrams or otherwise, the star and mesh network topologies Describe the differences between a local area network and a wide area network such as the Internet Describe the nature of the Internet as a worldwide collection of computer networks Identify different transmission media		**Network security and systems software** List some of the threats posed to networks, including malware and phishing Explain briefly what is meant by phishing and how to keep data safe from phishing attacks List precautions which can be taken to keep data safe from hackers including anti-malware software, firewalls, user access levels, passwords and encryption List the functions of an operating system: user interface, memory management, multi-tasking, peripheral management, user and file management Explain briefly what is meant by memory management and multi-tasking Describe briefly the purpose of encryption, defragmentation and data compression software Describe different types of user interface		**Telium** **Programming Project** How to apply the 5 key principles section to solve complex problems and justifying their approach Using the T.I.M.E approach to programming a solution to a problem Build a text based game from the ground up. Creating the raw files needed to save the date in the game as well as the programmed script	
	Assessment 1 Format:	Questions adapted from OCR specifications including short form answers and 8/15 mark long form answers.		Assessment 2 Format:	Questions adapted from OCR specifications including short form answers and 8/15 mark long form answers.	

	Autumn 1	Autumn 2	Spring 1	Spring 2	Summer 1	Summer 2
Y11	**Logic and languages** Digital Logic and Circuits Programming Languages and Translators Algorithm Testing and Validation		**Impacts of digital technology** List some ethical, legal, cultural or environmental issues in relation to a given scenario List some privacy issues in relation to a given scenario Choose from a given list, which Act is relevant to a particular scenario List one attribute and advantage of open source software and proprietary software		**Paper 1 and Paper 2 topic recalls** Retention of ALL units - Skills and Knowledge and applying them to certain types of questions Paper 1 - Written application of Computer Science Concepts Paper 2- Computation thinking and Logic	
	Mock Format:	Adapted OCR GCSE Exam Paper questions aligning with course content including short form answers and 8/15 mark long form answers.		Assessment 2 Format:	Adapted OCR GCSE Exam Paper questions aligning with course content including short form answers and 8/15 mark long form answers.	

L6th	**Programming** Programming Fundamentals and the Development Environment Structured Programming: Subroutines, Scope, and Arrays Advanced Paradigms: Recursion and Object-Oriented Programming **Computational Thinking** Problem Analysis and Abstraction Solution Design and Efficiency Advanced Computational Strategies	**Data Structures** Linear Data Structures: Stacks operate on a Last-In, First-Out (LIFO) basis Queues use a First-In, First-Out (FIFO) principle Lists, arrays, and tuples Non-Linear Data Structures Trees Graphs model networks Hash Tables **Data Types** Fundamental Data Types and Number Systems Representing Negative and Real Numbers Character Encoding and Bitwise Manipulation	**Data Structures** Linear Data Structures: Stacks operate on a Last-In, First-Out (LIFO) basis Queues use a First-In, First-Out (FIFO) principle Lists, arrays, and tuples Non-Linear Data Structures Trees Graphs model networks Hash Tables **Software Development** Software Development Methodologies and Testing Algorithm Design, Representation, and Analysis Programming Paradigms and Low-Level Architecture **Exchanging Data, SQL and Databases** Data Compression and Encryption Relational Database Design	**Exchanging Data, SQL and Databases** Data Compression and Encryption Relational Database Design Database Management with SQL **Systems Software** Core Operating System Functions and Resource Management The Software Ecosystem: From BIOS to Applications Code Translation and the Compilation Process **Boolean Algebra** Digital Logic Fundamentals: Gates, Circuits, and Boolean Expressions Logic Circuit Simplification Techniques Combinational and Sequential Logic Circuits **Components of a Computer** Core Processor Architecture and Operation	**Boolean Algebra** **Components of a Computer** Core Processor Architecture and Operation System Performance and Modern Processing The Memory and Storage Hierarchy **Algorithms** Core Searching and Sorting Algorithms Graph and Tree Traversal and Pathfinding Algorithms Algorithmic Complexity, Intractability, and Recursion **NEA - Project**	**Algorithms** Core Searching and Sorting Algorithms Graph and Tree Traversal and Pathfinding Algorithms Algorithmic Complexity, Intractability, and Recursion **NEA - Project**

	Autumn 1	Autumn 2	Spring 1	Spring 2	Summer 1	Summer 2
			Database Management with SQL	System Performance and Modern Processing The Memory and Storage Hierarchy		
	Assessment 1 Format:	Adapted OCR A level Paper on all content covered		Assessment 2 Format:	Adapted OCR A level Paper on all content covered. NEA project check-ins	
U6th	**Ethics** Societal and Cultural Impact Ethical Dilemmas and Artificial Intelligence Legal and Privacy Regulation **NEA - Project**	**Networks and HTML** Internet Structure and Communication Protocols Web Development and Server-Side vs. Client-Side Processing Network Security Threats and Defence Mechanisms **NEA - Project**	**NEA - Project**	**NEA - Project**	**NEA - Project**	**Study Leave**
	Mock Format:	Full mock paper on each of the three modules (adapted where necessary) and NEA project check-ins				

Key Vocabulary

KS3 Computer Science:

I. Core Concepts and Computational Thinking

- **Computational Thinking:** A way of thinking that uses concepts from computer science to solve problems, design systems, and understand human and machine intelligence. It equips pupils with creativity to understand and change the world.
- **Abstraction:** The process of removing unnecessary details from a problem or system to focus on the essential features. Pupils design, use, and evaluate computational abstractions that model real-world problems and physical systems.
- **Decomposition:** (Implied by computational thinking, though not explicitly defined at KS3 in the provided computing section, it's a foundational computational thinking principle mentioned in A Level and GCSE aims). It refers to breaking down a complex problem into smaller, more manageable parts.
- **Logic:** The use of logical reasoning and conditions to determine decision points and program flow. Students understand simple **Boolean logic** (AND, OR, NOT) and its uses in circuits and programming.
- **Algorithms:** A set of well-defined, ordered steps or instructions to solve a problem or accomplish a task. Pupils understand several key algorithms that reflect computational thinking, such as those for sorting and searching, and use logical reasoning to compare their utility.
- **Data Representation:** How data is stored and manipulated digitally within a computer system. It is a fundamental principle of computer science. Pupils understand how various data types (text, sounds, pictures) can be represented and manipulated digitally in the form of **binary digits**.

II. Programming and Software

- **Programming:** The process of creating instructions for a computer to follow. Pupils have practical experience of writing **computer programs** to solve problems.
- **Programming Languages:** Tools used to write computer programs. Pupils should use two or more, at least one of which is textual, to solve a variety of computational problems.
- **Data Structures:** Ways of organising data in memory. Pupils should make appropriate use of data structures such as **lists, tables, or arrays**.
- **Modular Programs:** Programs designed and developed using smaller, independent blocks of code called **procedures** or **functions**, which perform specific tasks.
    - **Procedures / Functions:** Reusable blocks of code that can be called upon to perform specific tasks within a program.

III. Computer Systems

- **Computer Systems:** Composed of **hardware** (physical components) and **software** (programs and data). Pupils understand how these components interact and communicate with one another and with other systems.
    - **Hardware Components:** The physical parts of a computer system.
    - **Software Components:** The programs and data that run on a computer system.
- **Instructions:** Commands that are stored and executed within a computer system.
- **Binary:** A number system with base 2, using only digits 0 and 1. Used to represent numbers and other data within a computer. Pupils understand binary

representation and can perform simple operations like binary addition and conversion between binary and decimal.

IV. Digital Literacy and Responsible Use

- **Information Technology (IT):** The use of computers and telecommunications to retrieve, store, transmit, and manipulate data. Pupils evaluate and apply IT, including new or unfamiliar technologies, analytically to solve problems.
- **Digitally Literate:** The ability to use, express oneself, and develop ideas through information and communication technology at a level suitable for the future workplace and as active participants in a digital world.
- **Digital Artefacts:** Creative projects or products developed using technology, requiring attention to trustworthiness, design, and usability.
- **Online Identity and Privacy:** Protecting one's personal information and presence when using technology online.
- **Technology Safely, Respectfully, Responsibly, and Securely:** Understanding how to use technology in a way that protects oneself and others, recognizing and reporting **inappropriate content, contact, and conduct**.

V. Cross-Curricular Connections (Mathematics and Science)

Computer Science at KS3 has deep links with mathematics and science.

- **Numerical and Mathematical Capability:** The ability to understand and apply numerical concepts, including the number system, place value, decimals, fractions, powers, and roots.
- **Algebra:** Using mathematical symbols to represent quantities and relationships, often involving expressions, equations, and inequalities.
- **Problem-Solving (Mathematical):** The ability to develop mathematical knowledge through solving problems, including breaking down more complex problems into a series of simpler steps.
- **Logical Reasoning:** The ability to compare algorithms and determine logical conditions.
- **Collecting, Presenting, and Analysing Data:** In science, pupils develop understanding of factors in collecting, recording, and processing data, and apply mathematical concepts to present and interpret observations using methods like tables and graphs.
- **Scientific Methods and Theories:** Understanding that scientific explanations are modified to account for new evidence and ideas, emphasizing objectivity, accuracy, precision, repeatability, and reproducibility in scientific enquiries.

<u>GCSE and ALevel Computer Science Vocabulary:</u>

I. Fundamental Principles and Computational Thinking

These terms represent core concepts that students are expected to understand and apply across various aspects of Computer Science:

- **Computational Thinking:** A mode of thought that extends beyond software and hardware, providing a framework for reasoning about systems and problems. It involves principles to define and refine problems.
- **Abstraction:** The process of removing unnecessary details from a problem or system to focus on the essential features. Students should understand its nature and need, and the difference between an abstraction and reality, being able to devise abstract models. It's a fundamental principle of computer science.
- **Decomposition:** Breaking down a complex problem into smaller, more manageable sub-problems. It's a fundamental principle of computer science.

- **Logic:** The use of logical reasoning and conditions to determine decision points and program flow. It's a fundamental principle of computer science.
- **Algorithms:** A set of well-defined, ordered steps or instructions to solve a problem or accomplish a task. Students should be able to design, create, and refine them. It's a fundamental principle of computer science.
- **Data Representation:** How data is stored and manipulated digitally within a computer system. It's a fundamental principle of computer science.

II. Computer Systems and Hardware

- **CPU (Central Processing Unit):** The purpose of the CPU is to execute instructions (the fetch-execute cycle).
    - **ALU (Arithmetic Logic Unit):** A common CPU component responsible for arithmetic and logic operations.
    - **Control Unit (CU):** A common CPU component that manages and coordinates the components of the CPU.
    - **Cache:** A small amount of very fast memory that stores frequently used instructions or data for quicker access by the CPU. Increasing cache size improves CPU performance.
    - **Registers:** Small, high-speed storage locations within the CPU that store data or addresses during the fetch-execute cycle (e.g., Program Counter, Accumulator, Memory Address Register (MAR), Memory Data Register (MDR), Current Instruction Register (CIR)).
    - **Fetch-Decode-Execute Cycle:** The sequence of actions performed by the CPU to retrieve an instruction, interpret it, and then carry it out.
    - **Clock speed, Number of cores:** Characteristics that affect CPU performance.
- **Embedded Systems:** Computer systems built into other devices, designed to perform one or a few dedicated functions with specific characteristics.
- **RAM (Random Access Memory):** Primary storage used to temporarily hold data and instructions that the CPU is actively using. It is volatile memory.
- **ROM (Read-Only Memory):** Primary storage that contains permanent, essential instructions (e.g., the BIOS) needed to start a computer system. It is non-volatile.
- **Virtual Memory:** An operating system technique where secondary storage is used as if it were RAM when the physical RAM is full, by transferring data between RAM and secondary storage.
- **Secondary Storage:** Non-volatile storage devices and media used for long-term storage of data (e.g., optical, magnetic, solid-state drives). Evaluated by capacity, speed, portability, durability, reliability, and cost.
- **Input/Output Devices:** Hardware components used to provide data to a computer (input) or display/present data from a computer (output).

III. Data Representation

- **Units of Data Storage (Bit, Nibble, Byte, KB, MB, GB, TB, PB):** Standard units for measuring data. A bit is the smallest unit (0 or 1), a nibble is 4 bits, a byte is 8 bits. Kilobyte (KB) is 1,000 bytes, Megabyte (MB) is 1,000 KB, Gigabyte (GB) is 1,000 MB, Terabyte (TB) is 1,000 GB, Petabyte (PB) is 1,000 TB. (Using 1,024 for conversions is also acceptable).
- **Binary:** A number system with base 2, using only digits 0 and 1, used to represent data inside a computer.
- **Hexadecimal:** A number system with base 16, used as a shorthand for binary (digits 0-9 and A-F).
- **Character Set (ASCII, Unicode):** A defined list of characters recognised by computer hardware and software, with corresponding binary codes to represent text. ASCII is an 8-bit character set (0-255), Unicode is a larger character set supporting more languages.
- **Pixels:** Individual points of colour that make up a digital image. Each pixel has a specific colour, represented by a specific binary code.
- **Metadata:** Data stored with an image that provides additional information about it (e.g., height, width, colour depth).
- **Colour Depth:** The number of bits used to represent the colour of a single pixel, affecting the quality and file size of an image.
- **Resolution:** The number of pixels per unit of area in an image, affecting its quality and file size.
- **Sample Rate (Sound):** Measured in Hertz (Hz), it is the number of samples taken per second when converting analogue sound to digital, affecting playback quality

and file size.

- **Bit Depth (Sound):** The number of bits available to store each sample of sound, affecting playback quality and file size.
- **Compression:** Reducing the size of a file.
    - **Lossy Compression:** Permanently removes some data to reduce file size, resulting in a loss of quality (e.g., JPEG, MP3).
    - **Lossless Compression:** Reduces file size without losing any data, so the original can be perfectly reconstructed (e.g., PNG, ZIP, Run Length Encoding, Dictionary Coding).

IV. Software and Development

- **Operating System (OS):** Systems software that manages computer hardware and software resources and provides common services for computer programs. Its functions include providing a user interface, memory management, multitasking, peripheral management, drivers, user management, and file management.
- **Virtual Machine:** An instance where software is used to take on the function of a physical machine, including executing intermediate code or running an operating system within another.
- **Utility Software:** System software that performs housekeeping tasks and additional functions not directly carried out by the operating system, such as encryption, defragmentation, and data compression.
- **Open Source Software:** Software whose source code is freely available and can be changed by anyone.
- **Closed Source Software (Proprietary Software):** Software where the source code is not accessible; it is typically purchased off-the-shelf and cannot be modified by the user.
- **Translators:** Software that converts programming code from one language to another.
    - **Compiler:** Translates an entire program into machine code before it is executed.
    - **Interpreter:** Translates and executes a program line by line.
    - **Assembler:** Translates assembly language into machine code.
- **Programming Paradigms:** Different fundamental styles of computer programming (e.g., procedural, object-oriented, assembly language).
- **Software Development Methodologies:** Structured approaches to software development (e.g., Waterfall lifecycle, Agile methodologies, Extreme Programming, Spiral Model, Rapid Application Development (RAD)).
- **IDE (Integrated Development Environment):** A software application that provides comprehensive facilities to computer programmers for software development, including editors, error diagnostics, a run-time environment, and translators.

V. Data Structures and Programming Concepts

- **Variables:** Named storage locations in a program used to hold data that can change during program execution.
- **Constants:** Named storage locations in a program used to hold data that remains fixed during program execution.
- **Operators:** Symbols or keywords that perform operations on values (operands) (e.g., arithmetic: +, -, *, /, MOD, DIV, ^; Boolean: AND, OR, NOT; comparison: ==, !=, <, <=, >, >=).
- **Programming Constructs:** Basic building blocks of programs that control the flow of execution.
    - **Sequence:** Instructions are executed in the order they appear.
    - **Selection (Branching):** Allows different code paths to be taken based on a condition (e.g., if/else, switch/case).
    - **Iteration (Looping):** Allows a block of code to be executed repeatedly (e.g., count-controlled for loop, condition-controlled while loop, do until loop).
- **Data Types:** Classification of data that tells the computer how the programmer intends to use the data (e.g., Integer, Real/Floating Point, Boolean, Character, String).

- **Casting:** Temporarily changing the data type of a variable or value.
- **String Manipulation:** Operations performed on strings, such as concatenating (joining), slicing (extracting a part), converting to uppercase/lowercase, or converting to ASCII.
- **Arrays:** Data structures that store a collection of items of the same data type under a single name, accessible via an index. Can be one-dimensional (1D) or two-dimensional (2D).
- **Data Structures (Advanced):** Ways of organising data in memory beyond simple arrays (e.g., Linked-list, Graph, Stack, Queue, Tree, Binary search tree, Hash table).
- **Sub Programs (Functions and Procedures):** Named blocks of code designed to perform a specific task, promoting modularity and reusability. Functions typically return a value, while procedures perform an action.
    - **Recursion:** A programming technique where a function or procedure calls itself to solve a problem, often by breaking it down into smaller instances of the same problem.
    - **Global and Local Variables:** Variables defined within a sub program are local (only accessible within that sub program), while global variables are accessible throughout the entire program.
- **Defensive Design:** Design considerations to ensure a program is robust and anticipates misuse or invalid data, including authentication and input validation.
    - **Authentication:** Confirming the identity of a user.
    - **Input Validation:** Checking that data entered by a user meets specific criteria before processing it.
- **Maintainability:** The ease with which a program can be understood, modified, and debugged, enhanced by practices like using sub programs, clear naming conventions, indentation, and commenting.
- **Testing:** The process of checking a program for errors.
    - **Iterative Testing:** Testing individual modules or sections of a program during its development.
    - **Final/Terminal Testing:** Testing the complete program at the end of the development process.
    - **Test Data (Normal, Boundary, Invalid/Erroneous):** Data used to test a program's functionality and robustness. Normal data is expected, boundary data is at the edge of validity, invalid data is of the correct type but should be rejected, and erroneous data is of the incorrect type and should be rejected.
    - **Syntax Errors:** Errors that break the grammatical rules of a programming language, preventing the program from running.
    - **Logic Errors:** Errors that cause a program to produce unexpected or incorrect output, even if it runs without crashing.

VI. Algorithms

- **Searching Algorithms:** Methods to find a specific item within a collection of data.
    - **Linear Search:** Checks each item in a list sequentially until the target is found or the list ends.
    - **Binary Search:** Efficiently finds an item in a sorted list by repeatedly dividing the search interval in half.
- **Sorting Algorithms:** Methods to arrange items in a list into a specific order.
    - **Bubble Sort:** Compares adjacent items and swaps them if they are in the wrong order, repeating until no more swaps are needed.
    - **Merge Sort:** Divides the list into halves, sorts each half recursively, and then merges the sorted halves.
    - **Insertion Sort:** Builds the final sorted array one item at a time by taking elements from the input and inserting them into their correct position in the sorted part.
- **Big O Notation:** A mathematical notation used to classify algorithms by how their run time or space requirements grow as the input size grows, measuring efficiency (e.g., constant, linear, polynomial, exponential, logarithmic complexity).

VII. Networks and Communications

- **LAN (Local Area Network):** A computer network that connects devices within a limited geographical area, like a home, office, or school.
- **WAN (Wide Area Network):** A computer network that spans a large geographical area, connecting multiple LANs (e.g., the Internet).
- **Client-Server Model:** A network architecture where clients request services from a server.
- **Peer-to-Peer Model:** A network architecture where all connected devices (peers) have equal capabilities and responsibilities, sharing resources directly with each other.
- **Router:** Network hardware that forwards data packets between computer networks.
- **Switch:** Network hardware that connects devices on a computer network by using packet switching to receive, process, and forward data to the destination device.
- **Wireless Access Point:** Hardware that allows wireless devices to connect to a wired network.
- **NIC (Network Interface Controller/Card):** Hardware that allows a computer to connect to a network.
- **The Internet:** A worldwide collection of computer networks, functioning as a global system of interconnected computer networks that uses the Internet protocol suite (TCP/IP) to link billions of devices.
- **DNS (Domain Name System):** A hierarchical distributed naming system for computers, services, or any resource connected to the Internet or a private network. It converts human-readable URLs into IP addresses.
- **The Cloud:** Remote service provision over the Internet (e.g., storage, software, processing).
- **Network Topologies (Star, Mesh):** The physical or logical arrangement of connected nodes in a network.
- **Protocols:** A set of rules for transferring data between devices on a network, ensuring consistent communication.
    - **TCP/IP (Transmission Control Protocol/Internet Protocol):** A suite of communication protocols used to interconnect network devices on the Internet.
    - **HTTP (Hyper Text Transfer Protocol) / HTTPS (Secure):** Protocols for transmitting hypermedia documents, such as HTML, over the Internet. HTTPS is the secure version.
    - **FTP (File Transfer Protocol):** A standard network protocol used for the transfer of computer files between a client and server on a computer network.
    - **POP (Post Office Protocol) / IMAP (Internet Message Access Protocol) / SMTP (Simple Mail Transfer Protocol):** Protocols for sending and receiving emails.
- **Standards:** Agreed-upon specifications or rules that ensure hardware and software from different manufacturers can interact and function together.
- **IP Addressing:** A numerical label assigned to each device connected to a computer network that uses the Internet Protocol for communication. Used for identification and location.
- **MAC Addressing:** A unique identifier assigned to a Network Interface Controller (NIC) for communications at the data link layer of a network segment.
- **Encryption:** The process of converting information or data into a code to prevent unauthorised access.
- **Firewalls:** Network security systems that monitor and control incoming and outgoing network traffic based on predetermined security rules.
- **Network Security Threats:** Various malicious activities or attacks that can compromise network integrity and data, such as Malware (malicious software), Social engineering (e.g., phishing), Brute-force attacks, Denial of Service (DoS) attacks, Data interception and theft, and SQL injection.
- **HTML (Hypertext Markup Language):** The standard markup language for documents designed to be displayed in a web browser.
- **CSS (Cascading Style Sheets):** A style sheet language used for describing the presentation of a document written in HTML.
- **JavaScript:** A programming language that enables interactive web pages.

VIII. Legal, Moral, Ethical, and Cultural Issues

- **Data Protection Act 2018:** Legislation concerning how personal data is collected, handled, and stored. (Supersedes Data Protection Act 1998).
- **Computer Misuse Act 1990:** Legislation against unauthorised access to computer material (hacking), unauthorised modification of computer material, and unauthorised access with intent to commit further offences.

- **Copyright, Designs and Patents Act 1988:** Legislation protecting original literary, dramatic, musical, or artistic works, including software.
- **Regulation of Investigatory Powers Act 2000 (RIPA):** Legislation governing the powers of public bodies to carry out surveillance and intercept communications.
- **Software Licences:** Legal agreements governing the use and distribution of software.
    - **Open Source Licence:** Grants users the right to use, study, change, and distribute the software and its source code to anyone and for any purpose.
    - **Proprietary Licence:** Restricts the use, modification, and distribution of software, typically requiring payment and not providing access to the source code.
- **Ethical Issues:** Moral principles that govern a person's behaviour or the conducting of an activity, particularly in the context of computer use (e.g., automated decision-making, artificial intelligence, environmental effects, censorship, privacy, piracy, offensive communications).

Suggested Reading List

KS3:

Computer Science for Fun - Queen Mary's Computer Science Magazine (available in IT1)

Computational fairy tales, by Jeremy Kubica

Coding for Beginners using Scratch (Usborne)

KS4:

Once upon an algorithm, Martin Erwig, 2017

Computational fairy tales, by Jeremy Kubica

Best Practices of Spell Design, by Jeremy Kubica

Logic, an introduction to elementary logic by Wilfred Hodges

Algorithms to live by: the computer science of human decisions. Christian and Griffiths, 2016

Nine Algorithms That Changed the Future: The Ingenious Ideas That Drive Today's Computers. John MacCormick 2012

Alevel:

FULL LIST: https://www.mrbarwick.com/readinglist

The new Turing omnibus, A Kee Dewdney, Palgrave Macmillan, 2003, ISBN 978-0805071665.

Computational Fairy Tales by Jeremy Kubica. ISBN: 978-1477550298

Computer Science: An Overview by J. Glenn Brookshear. ISBN: 978-0321544285

Code: The Hidden Language of Computer Hardware and Software by Charles Petzold. ISBN: 978-0735611313

Out of Their Minds by D Shasha and Cathy Lazere. ISBN: 978-3540979920

The Pattern on the Stone: The Simple Ideas That Make Computers Work by Daniel Hillis. ISBN: 978-0465025961

The Information: A History, a Theory, a Flood by James Gleick. ISBN: 978-0007225736

The Pleasures of Counting by Tom Kôrner. ISBN: 978-0521568234

The Code Book by Simon Singh. ISBN: 978-1857028898

Algorithmic Puzzles by Anany Levitin and Maria Levitin. ISBN: 978-0199740444