Computer Science

"The computer was born to solve problems that did not exist before."
— Bill Gates

Year 12

The Rationale: Computer Science is an ever evolving subject that engages and inspires students. They will learn theory of how computing devices work and how to develop software using high level programming languages.

	Autumn Term 1	Autumn Term 2	Spring Half Term 3	Spring Half Term 4	Summer Term 5	Summer Term 6
Curriculum Knowledge	(a) The Arithmetic and Logic Unit; ALU, Control Unit and Registers (Program Counter; PC, Accumulator; ACC, Memory Address Register; MAR, Memory Data Register; MDR, Current Instruction Register; CIR). Buses: data, address and control: how this relates to assembly language programs. (b) The Fetch-Decode-Execute Cycle; including its effects on registers. (c) The factors affecting the performance of the CPU: clock speed,	(a) The nature of abstraction. (b) The need for abstraction. (c) The differences between an abstraction and reality. (d) Devise an abstract model for a variety of situations. (a) Identify the inputs and outputs for a given situation. (b) Determine the preconditions for devising a solution to a problem. (c) The nature, benefits and drawbacks of caching. (d) The need for reusable program components.	(a) The need for, function and purpose of operating systems. (b) Memory Management (paging, segmentation and virtual memory). (c) Interrupts, the role of interrupts and Interrupt Service Routines (ISR), role within the Fetch-Decode-Execute Cycle. (d) Scheduling: round robin, first come first served, multi-level feedback queues, shortest job first and shortest remaining time. (e) Distributed, embedded, multi-tasking, multi-user	(a) Programming constructs: sequence, iteration, branching. (b) Recursion, how it can be used and compares to an iterative approach. (c) Global and local variables. (d) Modularity, functions and procedures, parameter passing by value and by reference. (e) Use of an IDE to develop/debug a program. (f) Use of object oriented techniques. (a) Features that make a problem solvable by	(a) Understand the waterfall lifecycle, agile methodologies, extreme programming, the spiral model and rapid application development. (b) The relative merits and drawbacks of different methodologies and when they might be used. (c) Writing and following algorithms. (a) Primitive data types, integer, real/floating point, character, string and Boolean. (b) Represent positive integers in binary. (c)	(a) Analysis and design of algorithms for a given situation. (b) The suitability of different algorithms for a given task and data set, in terms of execution time and space. (c) Measures and methods to determine the efficiency of different algorithms, Big O notation (constant, linear, polynomial, exponential and logarithmic complexity). (d) Comparison of the complexity of algorithms. (e) Algorithms for the

	number of cores, cache. (d) The use of pipelining in a processor to improve efficiency. (e) Von Neumann, Harvard and contemporary processor architecture. (a) The differences between and uses of CISC and RISC processors. (b) GPUs and their uses (including those not related to graphics). (c) Multicore and Parallel systems. (a) How different input, output and storage devices can be applied to the solution of different problems. (b) The uses of magnetic, flash and optical storage devices. (c) RAM and ROM. (d) Virtual storage	(a) Identify the components of a problem. (b) Identify the components of a solution to a problem. (c) Determine the order of the steps needed to solve a problem. (d) Identify sub-procedures necessary to solve a problem. (a) Identify the points in a solution where a decision has to be taken. (b) Determine the logical conditions that affect the outcome of a decision. (c) Determine how decisions affect flow through a program. (a) Determine the parts of a problem that can be tackled at the same time. (b) Outline the benefits and trade offs that might result from concurrent processing in a particular situation.	and Real Time operating systems. (f) BIOS. (g) Device drivers. (h) Virtual machines, any instance where software is used to take on the function of a machine, including executing intermediate code or running an operating system within another.	computational methods. (b) Problem recognition. (c) Problem decomposition. (d) Use of divide and conquer. (e) Use of abstraction. (f) Learners should apply their knowledge of: • backtracking • data mining • heuristics • performance modelling • pipelining • visualisation to solve problems.	Use of sign and magnitude and two's complement to represent negative numbers in binary. (d) Addition and subtraction of binary integers. (e) Represent positive integers in hexadecimal. (f) Convert positive integers between binary hexadecimal and denary. (g) Representation and normalisation of floating point numbers in binary. (h) Floating point arithmetic, positive and negative numbers, addition and subtraction. (i) Bitwise manipulation and masks: shifts, combining with AND, OR, and XOR. (j) How character sets (ASCII and UNICODE) are used to represent text.	main data structures, (stacks, queues, trees, linked lists, depth-first (post-order) and breadth-first traversal of trees). (f) Standard algorithms (bubble sort, insertion sort, merge sort, quick sort, Dijkstra's shortest path algorithm, A* algorithm, binary search and linear search).
Subject Skills	Able to code in assembly language. Know how to use	(a) Describe and justify the features that make the	(a) The nature of applications, justifying suitable	(a) Research the problem and solutions to similar	Plan a software development project	(a) Break down the problem into smaller parts

	code such as ADD, LDR, HLT, MOV, INT, END Be able to identify different processors and which are best for certain tasks. Be able to evaluate processor performance and how to increase performance by increasing the number of cores, increasing the clock speed or increasing the size of the cache.	problem solvable by computational methods. (b) Explain why the problem is amenable to a computational approach. (a) Identify and describe those who will have an interest in the solution explaining how the solution is appropriate to their needs (this may be named individuals, groups or persona that describes the target end user).	applications for a specific purpose. (b) Utilities. (c) Open source vs closed source. (d) Translators: Interpreters, compilers and assemblers. (e) Stages of compilation (lexical analysis, syntax analysis, code generation and optimisation). (f) Linkers and loaders and use of libraries.	problems to identify and justify suitable approaches to a solution. (b) Describe the essential features of a computational solution explaining these choices. (c) Explain the limitations of the proposed solution. (a) Specify and justify the solution requirements including hardware and software configuration (if appropriate). (b) Identify and justify measurable success criteria for the proposed solution.	using the Software Lifecycle, from Specification Requirements through to Design, Implementation, Testing and Maintenance. Create programs using primitive datatypes and the three programming constructs. Be able to convert binary integers, negative binary integers, fixed point binary and floating point binary.	suitable for computational solutions justifying any decisions made (a) Explain and justify the structure of the solution. (b) Describe the parts of the solution using algorithms justifying how these algorithms form a complete solution to the problem. (c) Describe usability features to be included in the solution. (d) Identify key variables / data structures / classes justifying choices and any necessary validation. (a) Identify the test data to be used during the iterative development and post development phases and justify the choice of this test data.
Tier 3 Vocabulary	Register Accumulator Program Counter Fetch Decode	Abstraction Decomposition Sequence Selection Iteration	Function Segmentation Virtual Interrupts Fetch	Sequence Selection Iteration Branching Recursion	Waterfall Lifecycle Methodology Spiral Rapid	Algorithm Execution Big O Notation Constant Linear

	Exexute Von Neumann CPU CISC RISC Multicore Parallel Magnetic Flash Optical Virtual Assembly	Function Procedure Concurrent Computational	Decode Execute Scheduling Distributed Embedded BIOS Driver Application Utility Translator Interpreter Compiler Assembler Lexical Syntax Optimisation Linkers Loaders	Iterative Global Local Modularity Function Procedure Modularity Parameter IDE Debug Object-oriented Computational Abstraction Decomposition Backtracking Data Mining Heuristics Pipelining Visualisation	Algorithm Integer Real Floating Boolean Binary Magnitude Twos-Complement Hexadecimal Exponent Mantissa Normalisation Bitwise	Polynomial Exponential Logarithmic Stacks Queues Trees Linked Lists Bubble Insertion Merge Dijkstra Decomposition Validation
How can you help your child engage with the content?	Talk about the inner workings of the devices you use today and how they are much smaller, yet faster that the first computer ever built.	Encourage problem solving and programming at home in high level languages.	Encourage Programming at home in high level languages.	Talk about their chosen A-Level project, who their client is and how they propose to build a program to solve the problem.	Talk about project management in places of work and the process companies have to go through in order to complete projects.	Encourage Programming at home in high level languages. Talk about the high level data structures and algorithms that solve problems.
Curriculum Opportunities	Computer Science trip to the Alan Turing Museum – Bletchley Park					
Career Links	Teaching and Education, Mentor, Computer Programmer, Games Developer, Games Designer, Web Designer, Artificial Intelligence https://nationalcareers.service.gov.uk/explore-careers					
Year 13 The Rationale: Computer Science is an ever evolving subject that engages and inspires students. They will learn theory of how computing devices work and how to develop software using high level programming languages.						
	Autumn Term 1	Autumn Term 2	Spring Half Term 3	Spring Half Term 4	Summer Term 5	Summer Term 6

Curriculum content	(a) Lossy vs Lossless compression. (b) Run length encoding and dictionary coding for lossless compression. (c) Symmetric and asymmetric encryption. (d) Different uses of hashing. (a) Relational database, flat file, primary key, foreign key, secondary key, entity relationship modelling, normalisation and indexing. (b) Methods of capturing, selecting, managing and exchanging data. (c) Normalisation to 3NF. (d) SQL – Interpret and modify (e) Referential integrity. (f) Transaction processing, ACID (Atomicity, Consistency, Isolation, Durability), record locking and redundancy.	(a) Arrays (of up to 3 dimensions), records, lists, tuples. (b) The following structures to store data: linked-list, graph (directed and undirected), stack, queue, tree, binary search tree, hash table. (a) Define problems using Boolean logic. (b) Manipulate Boolean expressions, including the use of Karnaugh maps to simplify Boolean expressions. (c) Use the following rules to derive or simplify statements in Boolean algebra: De Morgan's Laws, distribution, association, commutation, double negation. (d) Using logic gate diagrams and truth tables. (e) The logic associated with D type flip flops, half and full adders.	(a) Characteristics of networks and the importance of protocols and standards. (b) The internet structure: • The TCP/IP Stack. • DNS • Protocol layering. • LANs and WANs. • Packet and circuit switching. (c) Network security and threats, use of firewalls, proxies and encryption. (d) Network hardware. (e) Client-server and peer to peer (a) HTML, CSS and JavaScript. (b) Search engine indexing. (c) PageRank algorithm. (d) Server and client side processing.	(a) Need for and characteristics of a variety of programming paradigms. (b) Procedural languages. (c) Assembly language (including following and writing simple programs with the Little Man Computer instruction set) (d) Modes of addressing memory (immediate, direct, indirect and indexed). (e) Object-oriented languages with an understanding of classes, objects, methods, attributes, inheritance, encapsulation and polymorphism. (a) Define problems using Boolean logic. (b) Manipulate Boolean expressions, including the use of Karnaugh maps to simplify Boolean expressions. (c) Use the following rules to derive or simplify statements in Boolean algebra: De Morgan's Laws,	(a) The Data Protection Act 1998. (b) The Computer Misuse Act 1990. (c) The Copyright Design and Patents Act 1988. (d) The Regulation of Investigatory Powers Act 2000. The individual moral, social, ethical and cultural opportunities and risks of digital technology: • Computers in the workforce. • Automated decision making. • Artificial intelligence. • Environmental effects. • Censorship and the Internet. • Monitor behaviour. • Analyse personal information. • Piracy and offensive communications. • Layout, colour paradigms and character sets.	

				distribution, association, commutation, double negation. (d) Using logic gate diagrams and truth tables. (e) The logic associated with D type flip flops, half and full adders.		
Subject Skills	Database design including 1st, 2nd and 3rd Normal form. Entity relationship diagrams. Structured Query Language (SQL) to create tables, simple queries, parameter queries, cross table queries. Be able to write SQL to add, edit and delete records from a table.	How to create, traverse, add data to and remove data from the data structures mentioned above. (NB this can be either using arrays and procedural programming or an object-oriented approach).	Programming skills in web programming languages such as HTML, Javascript and CSS. Search engine indexing, PageRank Algorithm and Server and client side processing.	Object oriented programming. Be able to create classes, objects and instances. Be able to edit classes, objects and instances. Create truth tables for logic complex logic circuits that contain AND, OR, NOT and XOR gates. Create complex logic circuits from given Boolean Algebra expressions. Write complex Boolean Algebra expressions from given logic circuits	Be able to apply the appropriate legislation acts to given scenarios. Be able to evaluate the ethical, cultural, social and environmental issues and benefits and the impact that fast evolving technology has had on the world.	.
Tier 3 Vocabulary	Lossy Lossless Encoding Compression Hashing Database Primary Key	Array Tuple Linked-List Graph Stack Queue Tree	Network Protocol TCP/IP DNS LAN WAN Packet	NOT AND OR XOR ASCII Unicode Paradigms	Copyright Patent Computer Artificial-Intelligence Censorship Communications Paradigm	

	Foreign Key Secondary Key Entity Relationship Normalisation Indexing SQL Referential – Integrity Transaction ACID Redundancy	Hash-Table Procedural Object-Oriented	Switching Circuits Firewall Encryption Client Server Peer HTML CSS Javascript Pagerank	Procedural Assembly Object Oriented Attributes Inheritance Encapsulation Polymorhism	Character-Set	
How can you engage with your child?	Talk about real world databases such as the NHS, the Criminal Records Database or any other large company.	Encourage Programming at home in high level languages.	Encourage looking at the source code of websites you regularly visit.	Encourage Programming at home in high level languages.	Have discussions about fast evolving technology and how different things are nowadays compared to your childhood.	
Curriculum Opportunities	Computer Science trip to the Alan Turing Museum – Bletchley Park					
Career Links	Teaching and Education, Mentor, Computer Programmer, Games Developer, Games Designer, Web Designer, Artificial Intelligence https://nationalcareers.service.gov.uk/explore-careers					