Curriculum Design 2026-27

Computer Science

		Term1		Term2		Term3	
		Term 1.1	Term 1.2	Term 2.1	Term 2.2	Term 3.1	Term 3.2
12 & 13	Theme	*Computer Systems* *Algorithms & Programming*	*Computer Systems* *Algorithms & Programming*	*Computer Systems* *Algorithms & Programming*	*Computer Systems* *Algorithms & Programming*	*Computer Systems* *Algorithms & Programming*	*Computer Systems* *Algorithms & Programming*
	Concept	Components of a computer and their uses Legal, moral, cultural and ethical issues Elements of computational thinking Project	Components of a computer and their uses Exchanging data Data types, data structures and algorithms Project	Data Types Software and software development Data types, data structures and algorithms Project	Data Types Software and software development Data types, data structures and algorithms Project	Data Structures Exchanging data Data types, data structures and algorithms Project	Boolean Algebra Exchanging data Data types, data structures and algorithms Project
	Skills Knowledge	<u>Structure and function of the processor</u> • The Arithmetic and Logic Unit; ALU, Control Unit and Registers, Buses: data, address, and control • The factors affecting the performance of the CPU, clock speed, number of cores, cache • The fetch-decode-execute cycle, including its effect on registers, • The use of pipelining in a processor to improve efficiency • Von Neumann, Harvard, and contemporary processor architecture. <u>Computing related legislation</u> • The Data Protection Act 1998 • The Computer Misuse Act 1990 • The Copyright Design and Patents Act 1988 • The Regulation of Investigatory Powers Act 2000. <u>Moral and ethical Issues</u> • 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.	<u>Types of processors</u> • The differences between and uses of CISC and RISC processors, GPUs, and their uses (including those not related to graphics), Multicore and Parallel systems. <u>Input, output, and storage</u> • How different input, output and storage devices can be applied to the solution of different problems, The uses of magnetic, flash and optical storage devices, RAM and ROM, Virtual storage. <u>Networks</u> • Characteristics of networks and the importance of protocols and standards. • The internet structure: ○ The TCP/IP Stack. ○ DNS ○ Protocol layering. ○ LANs and WANs. ○ Packet and circuit switching. • Network security and threats, use of firewalls, proxies, and encryption. • Network hardware. • Client-server and peer to peer.	<u>Data Types</u> • Primitive data types, integer, real/floating point, character, string, and Boolean, • Represent positive integers in binary • Use of sign and magnitude and two's complement to represent negative numbers in binary, • Addition and subtraction of binary integers • Represent positive integers in hexadecimal <u>Systems Software</u> • The need for, function and purpose of operating systems • Memory management (paging, segmentation, and virtual memory), Interrupts, the role of interrupts and Interrupt Service Routines (ISR), role within the Fetch-Decode-Execute Cycle • Scheduling: round robin, first come first served, multi-level feedback queues, shortest job first and shortest remaining time • Distributed, embedded, multi-tasking, multi-user and Real Time operating systems • BIOS, device drivers, virtual machines	<u>Data Types</u> • Convert positive integers between binary hexadecimal and denary • Representation and normalisation of floating-point numbers in binary, • Floating point arithmetic, positive and negative numbers, addition, and subtraction • Bitwise manipulation and masks: shifts, combining with AND, OR, and XOR, • How character sets (ASCII and UNICODE) are used to represent text <u>Applications Generation</u> • The nature of applications, justifying suitable applications for a specific purpose • Utilities • Open-source vs closed source • Translators: Interpreters, compilers, and assemblers • Stages of compilation (lexical analysis, syntax analysis, code generation and optimisation), • Linkers and loaders and use of libraries. <u>Software Development</u> • Waterfall lifecycle • Agile methodologies • Extreme programming • Spiral model • Rapid application development <u>Types of Programming Language</u> • Need for and characteristics of a variety of programming paradigm • Procedural languages • Assembly • Modes of addressing memory • Object-oriented languages with an understanding of classes, objects, methods, attributes, inheritance, encapsulation and polymorphism.	<u>Data Structures</u> • Arrays (of up to 3 dimensions), records, lists, tuples • The following structures to store data: stack, queue, how to create, add data to and remove data from the data structures mentioned above (using arrays and procedural programming), the following structures to store data: linked-list, graph (directed and undirected), tree, binary search tree, hash 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). <u>Databases</u> • Relational database, flat file, primary key, foreign key, secondary key, entity relationship modelling, normalisation, and indexing. • Methods of capturing, selecting, managing, and exchanging data. • Normalisation to 3NF. • SQL – Interpret and modify. See appendix 5d. • Referential integrity. • Transaction processing, ACID (Atomicity, Consistency, Isolation, Durability), record locking and redundancy. <u>Compression, Encryption and Hashing</u> • Lossy vs Lossless compression run length encoding and dictionary coding for lossless compression, • Symmetric and asymmetric encryption, • Different uses of hashing. <u>Web Technologies</u> • HTML, CSS, and JavaScript. • Search engine indexing • PageRank algorithm • Server and client-side processing.	<u>Boolean Algebra</u> • Define problems using Boolean logic. See appendix 5d. • Manipulate Boolean expressions, including the use of Karnaugh maps to simplify Boolean expressions. • Use the following rules to derive or simplify statements in Boolean algebra: De Morgan's Laws, distribution, association, commutation, double negation. • Using logic gate diagrams and truth tables. See appendix 5d. • The logic associated with D type flip flops, half, and full adders. <u>Compression, Encryption and Hashing</u> • Lossy vs Lossless compression run length encoding and dictionary coding for lossless compression, • Symmetric and asymmetric encryption, • Different uses of hashing. <u>Web Technologies</u> • HTML, CSS, and JavaScript. • Search engine indexing • PageRank algorithm • Server and client-side processing.

		Computational Thinking
		• Thinking abstractly
		• Thinking ahead
		• Thinking procedurally
		• Thinking logically
		• Thinking concurrently
		
		Computational Methods
		• Features that make a problem solvable by computational methods
		• Problem recognition
		• Problem decomposition
		• Use of divide and conquer
		• Use of abstraction
		• Backtracking
		• Data mining
		• Heuristics
		• Performance modelling
		• Pipelining
		• Visualisation to solve problems.
		
		A-Level Project – Yr12
		Investigating programming languages
		
		A-Level Project – Yr13
		• Development
		  ○ Iterative development process
		  ○ Testing to inform development
			Algorithms
			• Analysis and design of algorithms for a given situation
			• The suitability of different algorithms for a given task and data set, in terms of execution time and space.
			• Measures and methods to determine the efficiency of different algorithms
			• Big O notation (constant, linear, polynomial, exponential and logarithmic complexity).
			
			A-Level Project – Yr12
			Programming a variety of challenges using a language of choice
			
			A-Level Project – Yr13
			• Development
			  ○ Iterative development process
			  ○ Testing to inform development
				Algorithms
				• Comparison of the complexity of algorithms.
				• Algorithms for the main data structures, (stacks, queues, trees, linked lists, depth-first (post-order) and breadth-first traversal of trees).
				• Standard algorithms (bubble sort, insertion sort, merge sort, quick sort, Dijkstra's shortest path algorithm, A* algorithm, binary search and linear search).
				
				A-Level Project – Yr12
				Programming a variety of challenges using a language of choice
				
				A-Level Project
				• Evaluation
				  ○ Testing to inform evaluation
				  ○ Success of the solution
					Programming Techniques
					• Modularity, functions and procedures, parameter passing by value and by reference.
					• Use of an IDE to develop/debug a program.
					• Use of object-oriented techniques
					
					A-Level Project – Yr12
					Deciding on a project idea
						Programming Technique
						• Modularity, functions and procedures, parameter passing by value and by reference.
						• Use of an IDE to develop/debug a program.
						• Use of object-oriented techniques
						
						A-Level Project – Yr12
						• Analysis
						  ○ Problem identification
						  ○ Stakeholders
						  ○ Research the problem
						  ○ Specify the proposed solution
							Programming Techniques
							• Modularity, functions and procedures, parameter passing by value and by reference.
							• Use of an IDE to develop/debug a program.
							• Use of object-oriented techniques
							
							A-Level Project – Yr12
							• Design
							  ○ Decompose the problem
							  ○ Describe the solution
							  ○ Describe the approach to testing

	Wider Curriculum						