Curriculum Map KS5 Computer Science

Year 12 - AS Level	Terms	Year 13 - A Level	Terms	Year 13 - A Level		Terms
Component 01: Computing Principles		**Component 1: Computer Systems**		**Component 03: Programming Project**		
Structure and Function of Processor	HT1	Networks	HT1	Analysis of the problem (10 marks)		HT1
Types of Processor	HT1	Web Technologies	HT1		Problem identification	Year 12 - HT6
Input, Output and storage	HT1	Compression, Encryption and Hashing	HT2		Stakeholders	Year 12 - HT6
Operating Systems	HT2	Databases & SQL	HT2		Research the problem	
Applications Generation	HT2	Computing Related Legislation	HT2		Specify the proposed solution	
Introduction to Programing	HT2	Ethic, moral and cultural issues	HT2	Design of the solution (15 marks)		HT1/ HT2
Data Types	HT3	Data Types including Binary Arithmatic	HT3 - Re-Visit		Decompose the problem	
Data Structures	HT3	Data Structures	HT3 - Re-Visit		Describe the solution	
Boolean Algebra	HT4	Boolean Algebra	HT3 - Re-Visit		Describe the approach to testing	
Databases	HT5	Structure and Function of Processor	HT4 - Re-Visit	Developing the solution (25 marks)		
Computing Related Legislation	HT5	Types of Processor	HT4 - Re-Visit		Iterative development process	HT2
Ethic, moral and cultural issues	HT5	Input, Output and storage	HT4 - Re-Visit		Testing to inform development	HT3/ HT4
Networks	HT6	Systems Software	HT4 - Re-Visit	Evaluation (20 marks)		HT4
Web Technologies	HT6	Applications Generation & Software Development	HT4 - Re-Visit		Testing to inform evaluation	
		Types of Programming Language	HT4 - Re-Visit		Success of the solution	
		Revision/ Exam practice	HT5		Describe the final product	
					Maintenance and development	
Component 02: Algorithms and Problem Solving		**Component 02: Algorithms and Problem Solving**				
Thinking Abstractly	HT1	Thinking Abstractly	HT1 - Re-visit			
Thinking Ahead	HT1	Thinking Ahead	HT1 - Re-visit	Programming Project to be finalised before 20th April.		
Thinking Procedurally (revisit with Programming Tech).	HT1	Thinking Logically	HT1 - Re-visit			
Thinking Logically	HT1	Thinking Concurrently	HT1			
Algorithms (revisit with programming Tech).	HT2	Thinking Procedurally with Programming Tech.	HT2 - Re-Visit			
Programming Techniques	HT3/ HT4	Programming Techniques	HT2			
Software Development	HT5/ HT6	Algorithms	HT2 & HT3			
		Computation Methods	HT4 & HT5			
Start on Component 03	HT6 - 6 lessons					
Problem identification						
Stakeholders						

Component 01 Topics

Topic	Sub Topic	Common content	AS Level	A Level
Structure and Function of Processor	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			
	The Fetch-Decode-Execute Cycle; including its effects on registers.			
	The factors affecting the performance of the CPU: clock speed, number of cores, cache.			
	The use of pipelining in a processor to improve efficiency			
	Von Neumann, Harvard and contemporary processor architecture.			
Types of Processor	The differences between and uses of CISC and RISC processors.			
	GPUs and their uses (including those not related to graphics).			
	Multicore and Parallel systems.			
Input, Output and storage	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.			
Systems Software (A Level)	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.			

Operating Systems (AS Level)	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, 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.			
Applications Generation	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.			
Software Development	Understand the waterfall lifecycle, agile methodologies, extreme programming, the spiral model and rapid application development.			
	The relative merits and drawbacks of different methodologies and when they might be used.			
	Writing and following algorithms.			
	Different test strategies, including black and white box testing and alpha and beta testing			
	Test programs that solve problems using suitable test data and end user feedback, justify a test strategy for a given situation.			
	Need for and characteristics of a variety of programming paradigms.			

Types of Programming Language	Procedural languages: • program flow • variables and constants • procedures and functions • arithmetic, Boolean and assignment operators • string handling • file handling.			
	Assembly language (including following and writing simple programs with the Little Man Computer instruction set).			
	Modes of addressing memory (immediate, direct, indirect and indexed).			
	Object-oriented languages with an understanding of classes, objects, methods, attributes, inheritance, encapsulation and polymorphism.			
Compression, Encryption and Hashing	Lossy vs. Lossless compression.			
	Run length encoding and dictionary coding for lossless compression.			
	Symmetric and asymmetric encryption.			
	Different uses of hashing.			
Databases	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.			
	Referential integrity.			
	Transaction processing, ACID (Atomicity, Consistency, Isolation, Durability), record locking and redundancy.			
Networks	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.			
Web Technologies	HTML, CSS and JavaScript.			
	Search engine indexing.			
	PageRank algorithm.			
	Server and client side processing.			
Data Types	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.			
	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.			
	Positive and negative real numbers using normalised floating point representation			
	How character sets (ASCII and UNICODE) are used to represent text.			
Boolean Algebra	Define problems using boolean logic.			
	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.			
	The logic associated with D type flip flops, half and full adders.			
Computing related legislation	The Data Protection Act 1998.			
	The Computer Misuse Act 1990.			
	The Copyright Design and Patents Act 1988.			
	The Regulation of Investigatory Powers Act 2000.			
Moreal and Ethical Issues	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.			

Component 02 Topics

Topic	Sub Topic	Common content	AS Level	A Level
Thinking Abstractly	The nature of abstraction.			
	The need for abstraction.			
	The differences between an abstraction and reality.			
	Devise an abstract model for a variety of situations.			
Thinking Ahead	Identify the inputs and outputs for a given situation.			
	Determine the preconditions for devising a solution to a problem.			
	The nature, benefits and drawbacks of caching.			
	The need for reusable program components.			
Thinking Procedurally	Identify the components of a problem.			
	Identify the components of a solution to a problem.			
	Determine the order of the steps needed to solve a problem.			
	Identify sub-procedures necessary to solve a problem.			
Thinking Logically	Identify the points in a solution where a decision has to be taken.			
	Determine the logical conditions that affect the outcome of a decision.			
	Determine how decisions affect flow through a program.			
Thinking Concurrently	Determine the parts of a problem that can be tackled at the same time.			
	Outline the benefits and trade offs that might result from concurrent processing in a particular situation.			
Programming Techniques	Programming constructs: sequence, iteration, branching.			
	Recursion, how it can be used and compares to an iterative approach.			

	Global and local variables.			
	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.			
Software Development	Understand the waterfall lifecycle, agile methodologies, extreme programming, the spiral model and rapid application development.			
	The relative merits and drawbacks of different methodologies and when they might be used.			
	Writing and following algorithms.			
	Different test strategies, including black and white box testing and alpha and beta testing			
	Test programs that solve problems using suitable test data and end user feedback, justify a test strategy for a given situation.			
Computational Methods	Features that make a problem solvable by computational methods.			
	Problem recognition.			
	Problem decomposition.			
	Use of divide and conquer.			
	Use of abstraction.			
	Learners should apply their knowledge of: • backtracking • data mining • heuristics • performance modelling • pipelining • visualisation to solve problems.			
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.			

	Standard algorithms (bubble sort, insertion sort, binary search and linear search).			
	Standard algorithms (quick sort, Dijkstra's shortest path algorithm, A* algorithm, binary search).			
	Implement bubble sort, insertion sort.			
	Implement binary and linear search.			
	Representing, adding data to and removing data from queues and stacks.			
	Measures and methods to determine the efficiency of different algorithms, Big O notation (constant, linear, polynomial, exponential and logarithmic complexity).			
	Algorithms for the main data structures, (stacks, queues, trees, linked lists, depth-first (post-order) and breadth-first traversal of trees).			
	Comparison of the complexity of algorithms.			
	Compare the suitability of different algorithms for a given task and data set.			

Component 03 Topics

Topic	Sub Topic
Problem Identification	Describe and justify the features that make the problem solvable by computational methods.
	Explain why the problem is amenable to a computational approach.
Stakeholders	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).
Research the Problem	Research the problem and solutions to similar problems to identify and justify suitable approaches to a solution.
	Describe the essential features of a computational solution explaining these choices.
	Explain the limitations of the proposed solution.
Specify the Proposed Solution	Identify the points in a solution where a decision has to be taken.
	Determine the logical conditions that affect the outcome of a decision
	Determine how decisions affect flow through a program.
Decompose the Problem	Break down the problem into smaller parts suitable for computational solutions justifying any decisions made.
Describe the solution	Explain and justify the structure of the solution
	Describe the parts of the solution using algorithms justifying how these algorithms form a complete solution to the problem.
	Describe usability features to be included in the solution.
	Identify key variables / data structures / classes justifying choices and any necessary validation.

Describe the approach to testing	Understand the waterfall lifecycle, agile methodologies, extreme programming, the spiral model and rapid application development.
	The relative merits and drawbacks of different methodologies and when they might be used.
	Writing and following algorithms.
	Different test strategies, including black and white box testing and alpha and beta testing.
	Test programs that solve problems using suitable test data and end user feedback, justify a test strategy for a given situation.
Iterative Development Process	Provide annotated evidence of each stage of the iterative development process justifying any decision made. Provide annotated evidence of prototype solutions justifying any decision made.
Testing to inform development	Provide annotated evidence for testing at each stage justifying the reason for the test. Provide annotated evidence of any remedial actions taken justifying the decision made.
Testing to inform evaluation	Provide annotated evidence of testing the solution of robustness at the end of the development process. Provide annotated evidence of usability testing (user feedback).
Success of the solution	Use the test evidence from the development and post development process to evaluate the solution against the success criteria from the analysis.
Describe the final product	Provide annotated evidence of the usability features from the design, commenting on their effectiveness.
Maintenance and development	Discuss the maintainability of the solution. Discuss potential further development of the solution.