

Curriculum Map KS5: Computer Science (OCR H446)

	Topic	Key Knowledge <i>What will students know by the end of this topic?</i>	Key Skills <i>What skills will students have developed by the end of this topic?</i>	Assessment Opportunities <i>How will student progress be measured? What are the key assessment pieces?</i>
Half Term 1	1.2.4b Types of Programming Language 2.2.1 Programming Techniques	1.2.4 Types of Programming Language (b) Procedural languages - Constructs: sequence, selection, iteration. - Data types, type casting, variables, scope. - Subroutines, parameter passing, modular design. 1D/2D arrays, tuples, records. - String slicing & basic file handling (read/write). 2.2.1 Programming Techniques (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.		Project Euler programming tasks HW notes Summary notes of topics PPQs Topic tests MCQs
	1.1.1 Structure and function of the processor 1.1.2 Types of processor 1.1.3 Input, output and storage	1.1.1 Structure and Function of the Processor (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, number of cores, cache. (d) The use of pipelining in a processor to improve efficiency. (e) Von Neumann, Harvard and contemporary processor architecture. 1.1.2 Types of Processor (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 1.1.3 Input, Output, and Storage Devices (a) How different input, output and storage devices can be applied to the solution of different problems.	1.1.1 Structure and Function of the Processor (a) Explain how data, addresses, and control signals move between registers and memory during execution. (b) Describe the logical sequence of events during the Fetch-Decode-Execute cycle and how the status of key registers changes at each stage. (c) Evaluate how changes to clock speed, cache capacity/levels, and core counts impact overall CPU processing performance (d) Explain how overlapping the FDE stages of multiple instructions improves CPU efficiency and describe why branch instructions force the pipeline to flush. (e) Compare the features, advantages, and drawbacks of Von Neumann, Harvard, and contemporary architectures for given application contexts 1.1.2 Types of Processor (a) Compare RISC and CISC architectures and justify processor selection for specific scenario contexts (b) Explain how the highly parallel architecture of GPUs differs from CPUs and justify their use in non-graphical applications (c) Distinguish between multi-core CPUs and parallel systems, explaining how parallel processing speeds up tasks alongside its limitations 1.1.3 Input, Output, and Storage Devices (a) Recommend and justify suitable combinations of input, output, and storage devices to fulfil specific user or system requirements.	

	Topic	Key Knowledge <i>What will students know by the end of this topic?</i>	Key Skills <i>What skills will students have developed by the end of this topic?</i>	Assessment Opportunities <i>How will student progress be measured? What are the key assessment pieces?</i>
		(b) The uses of magnetic, flash and optical storage devices. (c) RAM and ROM. (d) Virtual storage	(b) Compare magnetic, flash, and optical storage technologies across technical criteria to justify choices for given scenarios. (c) Explain the functional roles and volatility of RAM and ROM in a system, detailing their necessity during system startup and execution. (d) Explain the concept of virtual storage and evaluate its advantages and limitations	
Half Term 2	1.4.2 Data Structures	1.4.2 Data Structures (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. (c) 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).		
	2.1.1 – 2.1.5 Computational Thinking 1.2.3 Software Development 1.2.1a-d Systems Software	2.1.1 Thinking abstractly (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. 2.1.2 Thinking ahead (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. 2.1.3 Thinking procedurally (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. 2.1.4 Thinking logically (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. 2.1.5 Thinking concurrently		

	Topic	Key Knowledge <i>What will students know by the end of this topic?</i>	Key Skills <i>What skills will students have developed by the end of this topic?</i>	Assessment Opportunities <i>How will student progress be measured? What are the key assessment pieces?</i>
		(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. 1.2.3 Software Development (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. 1.2.1 Systems Software (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.		
Half Term 3	1.2.4e Types of Programming Language	1.2.4 Types of Programming Language (e) Object-oriented languages with an understanding of classes, objects, methods, attributes, inheritance, encapsulation and polymorphism. 2.2.1 Programming Techniques (f) Use of object oriented techniques.	•	
	2.2.1 Programming Techniques 1.2.1e-h Systems Software 1.2.2 Applications Generation	1.2.1 Systems Software (e) Distributed, embedded, multi-tasking, multi-user 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 1.2.2 Applications Generation (a) The nature of applications, justifying suitable applications for a specific purpose.	•	

	Topic	Key Knowledge <i>What will students know by the end of this topic?</i>	Key Skills <i>What skills will students have developed by the end of this topic?</i>	Assessment Opportunities <i>How will student progress be measured? What are the key assessment pieces?</i>
		(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.		
Half Term 4	3.1 Analysis of the Problem	3.1.1 Problem identification (a) Describe and justify the features that make the problem solvable by computational methods. (b) Explain why the problem is amenable to a computational approach. 3.1.2 Stakeholders (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). 3.1.3 Research the problem (a) Research the problem and solutions to similar 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. 3.1.4 Specify 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.	•	
	1.2.4a, c-d Types of Programming Language 1.3.3 Networks	1.2.4 Types of Programming Language (a) Need for and characteristics of a variety of programming paradigms. (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). 1.3.3 Networks (a) Characteristics of networks and the importance of protocols and standards.	•	

	Topic	Key Knowledge <i>What will students know by the end of this topic?</i>	Key Skills <i>What skills will students have developed by the end of this topic?</i>	Assessment Opportunities <i>How will student progress be measured? What are the key assessment pieces?</i>
		(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.		
Half Term 5	3.2 Design the Solution	3.2.1 Decompose the problem (a) Break down the problem into smaller parts suitable for computational solutions justifying any decisions made. 3.2.2 Describe the solution (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. 3.2.3 Describe the approach to testing (a) Identify the test data to be used during the iterative development and post development phases and justify the choice of this test data.	•	
	1.3.4 Web Technologies Targeted Revision and Review	1.3.4 Web Technologies (a) HTML, CSS and JavaScript. See appendix 5d. (b) Search engine indexing. (c) PageRank algorithm. (d) Server and client side processing.	•	
Half Term 6	3.3 Developing the Solution	3.3.1 Iterative Development Process (a) Provide annotated evidence of each stage of the iterative development process justifying any decision made. (b) Provide annotated evidence of prototype solutions justifying any decision made. 3.3.2 Testing to Inform Development (a) Provide annotated evidence for testing at each stage justifying the reason for the test. (b) Provide annotated evidence of any remedial actions taken justifying the decision made.	•	

	Topic	Key Knowledge <i>What will students know by the end of this topic?</i>	Key Skills <i>What skills will students have developed by the end of this topic?</i>	Assessment Opportunities <i>How will student progress be measured? What are the key assessment pieces?</i>
	1.5.2 Moral and Ethical Issues	1.5.2 Moral 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.	•	
Half Term 7	3.3 Developing the Solution 3.4 Evaluation	3.3.1 Iterative Development Process (a) Provide annotated evidence of each stage of the iterative development process justifying any decision made. (b) Provide annotated evidence of prototype solutions justifying any decision made. 3.3.2 Testing to Inform Development (a) Provide annotated evidence for testing at each stage justifying the reason for the test. (b) Provide annotated evidence of any remedial actions taken justifying the decision made. 3.4.1 Testing to Inform Evaluation (a) Provide annotated evidence of testing the solution of robustness at the end of the development process. (b) Provide annotated evidence of usability testing (user feedback).	•	
	2.3.1 Algorithms	2.3.1 Algorithms (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 main data structures, (stacks, queues, trees, linked lists, depth-first (post-order) and breadth-first traversal of trees).	•	

	Topic	Key Knowledge <i>What will students know by the end of this topic?</i>	Key Skills <i>What skills will students have developed by the end of this topic?</i>	Assessment Opportunities <i>How will student progress be measured? What are the key assessment pieces?</i>
		(f) Standard algorithms (bubble sort, insertion sort, merge sort, quick sort, Dijkstra's shortest path algorithm, A* algorithm, binary search and linear search).		
Half Term 8	3.4 Evaluation 1.4.1a-f Data Types	3.4.2 Success of the Solution (a) Use the test evidence from the development and post development process to evaluate the solution against the success criteria from the analysis. 3.4.3 Describe the Final Solution (a) Provide annotated evidence of the usability features from the design, commenting on their effectiveness. 3.4.4 Maintenance and Development (a) Discuss the maintainability of the solution. (b) Discuss potential further development of the solution. 1.4.1 Data Types (a) Primitive data types, integer, real/floating point, character, string and Boolean. (b) Represent positive integers in binary. (c) 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.	•	
	1.5.1 Computing Related Legislation 1.3.2 Databases	1.5.1 Computing Related Legislation (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. 1.3.2 Databases (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.	•	

	Topic	Key Knowledge <i>What will students know by the end of this topic?</i>	Key Skills <i>What skills will students have developed by the end of this topic?</i>	Assessment Opportunities <i>How will student progress be measured? What are the key assessment pieces?</i>
Half Term 9	1.4.1g-j Data Types	1.4.1 Data Types (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.	•	
	1.3.1 Compression, Encryption and Hashing Targeted revision	1.3.1 Compression, Encryption and Hashing (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.	•	
Half Term 10	1.4.3 Boolean Algebra	1.4.3 Boolean Algebra (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.	•	
	2.2.2 Computational Methods	2.2.2 Computational Methods (a) Features that make a problem solvable by 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.	•	
Half Term 11	Revision			

