KS5 Curriculum Map – Computer Science:

Topic	**Substantive Knowledge** This is the specific, factual content for the topic, which should be connected into a careful sequence of learning.	**Disciplinary Knowledge (Skills)** This is the action taken within a particular topic in order to gain substantive knowledge.	**Assessment Opportunities** What assessments will be used to measure student progress?
Data Types	• Primitive data types, binary and hexadecimal • ASCII and Unicode • Binary arithmetic • Floating point arithmetic • Bitwise manipulation and masks	• Students will be able to convert to different number systems (binary, hexadecimal and denary) • Students are able to represent negative numbers using sign and magnitude and two's complement • Perform binary addition and subtraction • Representation of normalisation of floating-point numbers • Perform floating point arithmetic • Apply bitwise manipulation and masks, combining with AND, OR and XOR • How to represent characters sets (ASCII and UNICODE)	• Students will complete a range of homeworks to test the skills learnt • Students will complete worksheets and questions from the OCR text book • End of unit test to test students understanding of the whole topic
Boolean Algebra	• Logic fates and truth tables • Simplifying Boolean expressions • Karnaugh maps • Adders and D-type flip-flops	• 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 • Understand the logic for half and full adders	• Apply problem solving skills to create logic gate circuits for real world scenarios • Worksheets to tests student's ability to simplify Boolean expressions • Create half and full adders using logic.ly and breadboards in lessons • Homeworks to consolidate students understanding

			• End of unit topic test
Data structures	• Arrays, tuples and records • Queues • Lists and linked lists • Stacks • Hash tables • Graphs • Trees	• Arrays (of up to 3 dimensions), records, lists, tuples – how to create and iterate through in a high-level programming language • Create a linked-list and how to insert, delete from a linked list • Graph (directed and undirected) and how to traverse through a graph • Implementation and operations of a stack and how they are used in functions • Trees and the key concepts and how to perform a range of traversals, binary search tree, • Hash tables and hashing algorithms with the use of dictionaries	• Students will be assessed in Python by completing a range of programming tasks to create the data structures using OOP • Worksheets to test student's knowledge and understanding • Homeworks given for each data structure • End of unit topic test
Programming Techniques	• Programming Basics • Selection • Iteration • Subroutines • Recursion • Object-Oriented Programming	• use arithmetic operations and Boolean operations NOT, AND and OR • use functions and library subroutines including random number generation • know how to define and call a subroutine (procedure or function) with parameters • construct algorithms using one-dimensional arrays • describe what is meant by recursion • define the OOP terms class, object, method, attribute, inheritance, encapsulation and polymorphism • draw an inheritance diagram • describe features of an IDE which are useful in developing and debugging a program	• Students will be assessed in their construction of classes and sub-classes in Python. • While students will develop classes in Python, they will develop their understanding of constructor methods and how to interpret them in Pseudocode (OCR Reference Language). • Students will complete a range of homeworks to test the skills learnt • Students will complete worksheets and questions from the OCR text book

		• write a pseudocode solution for a problem involving iteration and selection (branching) • use structured programming techniques and write their own subroutines with parameters • construct algorithms using two-dimensional arrays • use local and global variables in subroutines • trace through a recursive algorithm • compare iterative and recursive algorithms for solving a problem • complete given pseudocode for an object-oriented program • write complex algorithms involving data structures, subroutines and file-handling • interpret complex algorithms and determine the output • explain why using local variables makes a program easier to maintain • distinguish between passing parameters by value and by reference • write a recursive algorithm to solve a problem • use object-oriented programming techniques to solve problems	• End of unit topic test.
Exchanging Data	• Compression and Encryption • Database Concepts • Relational Databases and Normalisation • Introduction to SQL • Defining and Updating Tables using SQL • Transaction Processing	• explain the difference between lossy and lossless compression and list advantages and disadvantages of each • define the terms relational database, foreign key, secondary key, entity • draw a simple entity relationship diagram involving three or four entities • state the properties of a database in Third Normal Form • interpret a simple SQL statement • list methods of capturing data for input to a database	• Students will create, interpret and explain SQL statements. • Students will reduce the duplication of data and use normalisation to allow for consistent data across a large database. • Students will use SQL to create, modify and delete data/databases

		• explain the differences between asymmetric and symmetric encryption • explain the use of hashing to encrypt data • draw a complex entity relationship diagram involving several entities • normalise a database to third normal form • list the advantages of a normalised database • describe methods of capturing, selecting, managing and exchanging data • Describe what is meant by redundancy • Explain what is meant by referential integrity • use SQL to modify a database • describe what is meant by transaction processing and ACID	• Students will complete a range of homeworks to test the skills learnt • Students will complete worksheets and questions from the OCR text book • End of unit test
Computational Thinking	• Thinking Abstractly • Thinking Ahead • Thinking Procedurally • Thinking Logically, Thinking Concurrently • Problem Recognition • Problem Solving	• explain the differences between an abstraction and reality • describe the need for reusable program components • identify the inputs and outputs for a given situation • interpret simple algorithms to describe their purpose • give an example of how caching is used in a computer system • determine the preconditions for devising a solution to a problem • describe the nature, benefits and drawbacks of caching • identify the components of a problem and its solution • determine the order of steps needed to solve a problem • determine the logical conditions that affect the outcome of a decision	• Students will complete a range of homeworks to test the skills learnt • Students will complete worksheets and questions from the OCR text book • End of Unit Test

		• describe the nature of and need for abstraction • devise an abstract model for a variety of situations • design algorithms to solve complex problems • hand trace a complex algorithm to say what it does • determine the parts of a problem that can be executed concurrently • outline the benefits and trade-offs that might result from concurrent processing in a particular situation • apply techniques of backtracking, data mining, heuristics, performance modelling, pipelining and visualisation to the solution of problems	
Software Development	• Systems Analysis Methods • Writing and Following Algorithms • Programming Paradigms	• list the stages in the waterfall lifecycle model • name two other systems development models • name and describe different types of testing • write a pseudocode algorithm to solve a simple problem • use a trace table to trace through an algorithm • interpret simple algorithms to describe their purpose • list two features of a good algorithm • Define the term “programming paradigm” and give an example of two paradigms • define the terms object, class, method, attribute, inheritance • draw a simple inheritance diagram for a set of classes in an object-oriented approach	• Students will complete a range of homeworks to test the skills learnt • Students will complete worksheets and questions from the OCR text book • Students will differentiate between different Systems Analysis methods and refer to these in their programming project. • End of Unit test

		- describe agile methodologies, extreme programming, the spiral model and rapid application development - write pseudocode algorithms to solve problems - describe different programming paradigms, including procedural, and object-oriented paradigms - explain the terms encapsulation and polymorphism - distinguish between immediate, direct and indirect addressing modes in assembly language - describe the relative merits and drawbacks of different software development methodologies and when they might be used - design algorithms to solve complex problems - explain why different programming paradigms are suited to different applications and the advantages of each - describe and use four methods of addressing memory: immediate, direct, indirect and indexed	
Algorithms	- Analysis and design of algorithms - Searching algorithms - Bubble sort and insertion sort - Merge sort and quick sort - Graph traversal algorithms - Optimisation 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) - Comparison of the complexity of algorithms - Algorithms for the main data structures, (stacks, queues, trees, linked lists, depth-	- Programming tasks to create the algorithms previously mentioned - Trace tables to be able to trace through algorithms - Worksheets to tests student's ability to work out the time complexity - Worksheets to assess student's ability to describe algorithms - End of unit test

		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)	
Components of a computer	• Processor components • Processor performance • Types of processor • Input devices • Output devices • Storage devices	• Understand the functions of the following components: ALU, CU, PC, ACC, MAR, MDR, CIR • How data is sent between components via the address and data bus • The Fetch-Decode-Execute Cycle; including its effects on registers • The factors affecting the performance of the CPU: clock speed, number of cores, cache • How pipelining works • The difference between Von Neuman and Harvard architecture • The difference between CISC and RISC and how it is now impacting the market • How multicore and parallel systems work • GPUs and how they differ to CPU • Different types of technology used in secondary storage and their advantages and disadvantages • RAM, ROM and virtual storage and how swapping takes place	• Students will be assessed on their understanding of the FDE via the LMC • Complete a range of in class activities to identify whether parallel processing or multicore works better • Complete tasks in the OCR text book • Complete a range of homeworks to consolidate student's knowledge and understanding • End of unit test
Legal, moral, ethical and cultural issues	• Computing related legislation • Ethical, moral and cultural issues • Privacy and censorship	• Students understands the key factors of each of the following laws: The Data Protection Act 1998, The Computer Misuse Act 1990, The Copyright Design and Patents Act 1988, The Regulation of Investigatory Powers Act 2000 • To understand the impact that technology has on the following areas: • Computers in the workforce. • Automated decision making.	• Essay style writing questions • Group activities and presentation on different moral factors

		• Artificial intelligence. • Environmental effects. • Censorship and the Internet. • Monitor behaviour. • Analyse personal information. • Piracy and offensive communications. • Layout, colour paradigms and character sets	
Networks	• The Structure of the Internet • Internet Communication • HTML & CSS • JavaScript • Search Engine Indexing • Client-Server & Peer-to-Peer	• State the importance of protocols and standards • Describe the structure of the Internet • Explain the protocols used within the TCP/IP stack • Demonstrate DNS in action using an IP address within a web browser • Describe and identify examples of LANs and WANs • Explain packet switching • Provide examples of network threats and state methods to overcome these • Explain the function of a firewall • State the functions of a proxy server • Create a basic webpage using HTML and some CSS • Use JavaScript to make web form elements interactive and add validation • Describe the characteristics of the PageRank algorithm and state the factors that influence page ranking • Describe the processes at each layer of the TCP/IP stack • Explain the DNS resolution process • Explain packet switching in contrast to circuit switching • State the advantages of layering protocols in the TCP/IP stack • Explain, by use of example, the difference between client and server-side processing	• Students will complete a range of homeworks to test the skills learnt • Students will complete worksheets and questions from the OCR text book • Students will be assessed on their ability to create interactive and high-functioning web pages using HTML, CSS and JavaScript • End of Unit Test

		• Use sequence and selection statements in JavaScript with a range of data types including arrays • Describe how improved code quality can protect against networking vulnerabilities • Apply the PageRank algorithm using iterative steps	
Systems Software	• Functions of an Operating System • Types of Operating Systems • Nature of Applications • Programming Languages	• State the function and purpose of an operating system • Describe scheduling algorithms: round robin, first come first served, multi-level feedback queues, shortest job first and shortest remaining time • Describe distributed, embedded, multi-tasking, multi-user and real-time operating systems • Describe the function of BIOS and device drivers • Distinguish between systems software and applications software • Describe what is meant by a utility program and give examples • Be able to justify a suitable application for a specific purpose • Distinguish between open source and closed source software • State the roles of an assembler, compiler and interpreter • Describe the use of libraries • Describe memory management (paging, segmentation and virtual memory) • Describe the role of interrupts • Describe the need for processor scheduling algorithms • Explain the difference between compilation and interpretation, and describe situations when both would be appropriate • Describe what is meant by a virtual machine	• Students will complete a range of homeworks to test the skills learnt • Students will complete worksheets and questions from the OCR text book • End of Unit Test

		• Describe the stages of compilation: lexical analysis, syntax analysis, code generation and optimisation • Describe the function of linkers and loaders	