Curriculum Map Year 12 GCSE COMPUTER SCIENCE

Topic Name	**Term**	**Skills** developed with link to NC Subject content	Reflection on previous link in the curriculum	Progress to future link in the curriculum
Components of a computer	*Autumn HT1*	• Describe the function of the ALU and Control Unit • Describe the Fetch-Execute cycle and the role of the following registers: • Program counter • Accumulator • Memory Address Register • Memory Data Register • Current Instruction Register • Describe the factors affecting the performance of the CPU: clock speed, number of cores, cache • Understand the use of pipelining in a processor to improve efficiency • Describe von Neumann, Harvard and contemporary processor architecture • Describe the differences between, and uses of, CISC and RISC processors • Describe GPUs and their uses • Describe multicore and parallel systems • Describe different input devices • Explain how different input devices can be applied as a solution to different problems • Describe how different output devices can be applied as a solution of different problems • Describe the characteristics and uses of RAM and ROM • Understand what is meant by virtual storage • Describe the uses of magnetic, flash and optical storage devices	*Year 10: Systems architecture* *Year 10: Memory and storage*	*Examinations* *Year 12: Systems software*
System software	*Autumn HT2*	• Understand the function and purpose of an operating system • Describe memory management, interrupts, scheduling routines • Describe the role of interrupts and an Interrupt Service Routine (ISR) within the fetch-decode-execute cycle • Describe the need for processor scheduling algorithms • 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 BIOS, device drivers and virtual machines • 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 • Understand the role of an assembler, compiler and interpreter • Explain the difference between compilation and interpretation, and describe situations when both would be appropriate • Explain why an intermediate language such as bytecode is produced as the final output by some compilers and how it is subsequently used • Describe the stages of compilation: lexical analysis, syntax analysis, code generation and optimisation	*Year 10: Hardware and software* *Year 12: Components of a computer*	*Year 12: Software development* *Examinations* *NEA: Design, development, evaluation*

		• Describe the function of linkers and loaders • Describe the use of libraries		
Software development	*Autumn HT2 and HT3*	• Describe the waterfall lifecycle model, agile methodologies, extreme programming, the spiral model and rapid application development • Describe the relative merits and drawbacks of different methodologies and when they might be used • To understand the term 'algorithm' • To learn how to write and interpret algorithms using pseudocode • Understand the need for and characteristics of a variety of programming paradigms • Describe the features of procedural languages • Describe the features of declarative languages • Describe the features of object-oriented languages • Develop an understanding of classes, objects, methods, attributes, inheritance, encapsulation and polymorphism • Write and follow simple assembly language programs • Understand and apply immediate, direct, indirect and indexed addressing modes	*Year 10: Basic programming concepts* *Year 10: Advanced programming concepts* *Year 10: Robust and secure programming concepts* *Year 11: Python programming examination work*	*Year 13: Programming techniques* *Year 13: Algorithms* *Examination* *NEA: Analysis, Design, development, evaluation*
Exchanging data	*Spring HT3*	• Understand the difference between lossless and lossy compression • Explain run length encoding and dictionary based compression • Define symmetric and asymmetric encryption • Understand how and why hashing may be used to encrypt data • Explain the concept of a relational database • Define the terms flat file, entity, attribute, primary key, foreign key, secondary key, entity relationship modelling, referential integrity • Produce an entity relationship model for a simple scenario involving multiple entities • Describe the use of secondary keys and indexing • Describe the use of a hash table for index organisation • Normalise relations to third normal form • Understand why databases are normalised • Be able to use SQL to retrieve data from multiple tables of a relational database • Be able to interpret and modify SQL • Be able to use SQL to define a database table • Be able to use SQL to update, insert and delete data from multiple tables of a relational database • Describe methods of capturing, selecting, managing and exchanging data • Describe what is meant by transaction processing and ACID (Atomicity, Consistency, Isolation, Durability) • Describe what is meant by record locking and why it is necessary in a multi-user database • Describe what is meant by redundancy	*Year 10: Data representation* *Year 10: Advanced programming concepts* *Year 11: Python programming examination work* *Year 11: Relational databases and SQL*	*Examinations*
Networks and web technologies	*Spring HT4*	• Understand the structure of the Internet • Describe the term 'Uniform Resource Locator' in the context of networking • Explain the terms 'domain name' and 'IP address'	*Year 10: Computer networks, protocols and layers*	*Examinations*

		• Describe how domain names are organised • Understand the purpose and function of the Domain Name Server (DNS) system • Describe the characteristics of LANs and WANs • Describe circuit switching and packet switching • Understand the role of packet switching and routers • Understand the function of network hardware devices • Understand the importance of protocols and standards • Describe the roles of the four layers in the TCP/IP protocol stack • Be familiar with transferring files using FTP • Explain the role of an email server in sending and retrieving email • Discuss network security and threats • Discuss use of firewalls, proxies and encryption • Discuss worms, Trojans and viruses and the vulnerabilities that they exploit • To understand HTML and the role of HTML on the World Wide Web • To understand CSS and the role of CSS in Web Pages • To be familiar with various HTML and CSS tags and their functions • To use inline CSS directly within HTML files using the <style> tag, and with external style sheets • Be able to add HTML form controls to a web page • Explain the role of JavaScript inside web pages • Understand and follow JavaScript syntax • Write basic JavaScript code for a given scenario • Use JavaScript to change the content of HTML elements • Create output, including alert boxes using JavaScript • To understand how web pages are indexed by search engines • To understand the PageRank algorithm • To be able to interpret and apply the PageRank algorithm to a given scenario • To understand the client/server and peer-to-peer models • Describe situations where each model may be used • To understand client and server side processing • To identify the different uses of client and server side processing and describe situations when either may be most practical • To identify the advantages and disadvantages of client and server side processing		
Data types	*Spring HT5*	• List and define primitive data types • Represent positive integers in binary and hexadecimal • Convert between binary, hexadecimal and denary number systems • Define a bit as a 1 or a 0, and a byte as a group of eight bits • Know that 2n different values can be represented with n bits • Use names, symbols and corresponding powers of 2 for binary prefixes e.g. Ki, Mi • Differentiate between the character code of a denary digit and its pure binary representation • Describe how character sets (ASCII and Unicode) are used to represent text • Use sign and magnitude to represent negative numbers in binary • Use two's complement to represent negative numbers in binary • Add and subtract binary integers • Represent fractions in fixed point binary	*Year 10: Basic programming techniques* *Year 10: Advanced programming concepts* *Year 10: Robust and secure programming concepts* *Year 11: Python programming examination work* *Year 11: Algorithms* *Year 11: Boolean logic*	*Year 12: Data types* *Year 13: Programming techniques* *Year 13: Algorithms* *Examination revision*

		• Represent positive and negative numbers with a fractional part in floating point form • Normalise un-normalised floating point numbers with positive or negative mantissas • Add and subtract floating point numbers • Perform logical, arithmetic and circular shifts on binary data • Perform bitwise operations AND, OR and XOR • Use masks to manipulate bits		

Data structures	*HT5 and HT6*	• Be familiar with the concept of a data structure
		• Understand how data is represented and stored within different structures including
		  • arrays up to three dimensions
		  • lists
		  • tuples
		• Use 1-, 2- and 3-dimensional arrays in the design of solutions to simple problems
		• Understand the concept of an abstract data type
		• Be familiar with the concept and uses of a queue
		• Describe the creation and maintenance of data within a queue (linear, circular, priority)
		• Describe and apply the following to a linear, circular and priority queue
		  • add an item
		  • remove an item
		  • test for an empty queue
		  • test for a full queue
		• Explain how a list may be implemented as a static or dynamic data structure
		• Describe the linked list data structure
		• Show how to create, traverse, add data to and remove data from a linked list
		• Be familiar with the concept and uses of a stack
		• Be able to describe the creation and maintenance of data within a stack
		• Be able to describe and apply the following operations: push, pop, peek (or top), test for empty stack, test for full stack
		• Be able to explain how a stack frame is used with subroutine calls to store return addresses, parameters and local variables
		• Be familiar with a hash table and its uses
		• Be able to apply simple hashing algorithms
		• Know what is meant by a collision and how collisions are handled using rehashing
		• Be familiar with the concept of a dictionary
		• Be familiar with simple applications of a dictionary
		• Be aware of a graph as a data structure used to represent complex relationships
		• Be familiar with typical uses for graphs
		• Be able to explain the terms: graph, weighted graph, vertex/node, edge/arc, undirected graph, directed graph
		• Know how an adjacency matrix and an adjacency list may be used to represent a graph
		• Be able to compare the use of adjacency matrices and adjacency lists
		• Know that a tree is a connected, undirected graph with no cycles
			Year 10: Basic programming techniques
			Year 10: Advanced programming concepts
			Year 10: Robust and secure programming concepts
			Year 10: Python programming
			Year 11: Algorithms
			Year 12: Data types
				Year 12: Programming
				Year 13: Programming techniques
				Year 13: Algorithms
				Year 13: Computational thinking
				Examination revision

		• Know that a binary tree is a rooted tree in which each node has at most two children • Be familiar with typical uses for rooted trees		
Programming	*HT1 to HT5*	Thes topics are designed for students with no programming experience up to those with GCSE experience, progress and start points will be different depending on prior knowledge. These tasks are using to support future programming theory units and beginning the NEA • Structured programming • Selection, data types, counter-controlled iteration, condition-controlled iteration • Handling inputs • Using arrays and lists • Serial files	*Year 10: Basic programming techniques* *Year 10: Advanced programming concepts* *Year 10: Robust and secure programming concepts* *Year 10: Python programming* *Year 11: Algorithms* *Year 11: Python programming examination work* *Year 12: Data structures*	*Year 13: Programming techniques* *Year 13: Algorithms* *Examination revision* *NEA: Design, development, evaluation*
NEA: Analysis	*HT6*	• 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. • 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). • 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. • 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.	*Year 10: Basic programming techniques* *Year 10: Advanced programming concepts* *Year 10: Robust and secure programming concepts* *Year 10: Python programming* *Year 11: Algorithms* *Year 11: Python programming examination work* *Year 12: Software development*	*Year 13: Programming techniques* *Year 13: Algorithms* *Examination revision* *NEA: Design, development, evaluation*