Introduction

This mapping document highlights the overlapping content between the GCSE Computer Science and the International GCSE Computer Science. You may wish to use this mapping document to determine which resources designed specifically for the UK GCSE can support you in your teaching and planning for the International GCSE, including the dedicated textbooks. This mapping document demonstrates that 80% of the content is identical, 10% of the content is similar, and 10% of the content is different. Please note that the International GCSE and GCSE have different assessment methods.

Topic	Sub Topic	GCSE Reference	GCSE Learning Objective	Level of difference	International GCSE Reference	International GCSE Learning Objective
Problem Solving	Algorithms	1.1.1	understand what an algorithm is, what algorithms are used for and be able to interpret algorithms (flowcharts, pseudocode, written descriptions, program code)	Same	1.1.1	Understand what an algorithm is, what algorithms are used for and be able to interpret algorithms (flowcharts, pseudocode, written descriptions, program code).
		1.1.2	understand how to create an algorithm to solve a particular problem, making use of programming constructs (sequence, selection, iteration) and using appropriate conventions (flowchart, pseudo-code, written description, draft program code)	Same	1.1.2	Understand how to create an algorithm to solve a particular problem, making use of programming constructs (sequence, selection, iteration) and using appropriate conventions (flowchart, pseudocode, written description, draft program code).
		1.1.3	understand the purpose of a given algorithm and how an algorithm works	Same	1.1.3	Understand the purpose of a given algorithm and how an algorithm works.
		1.1.4	understand how to determine the correct output of an algorithm for a given set of data	Same	1.1.4	Understand how to determine the correct output of an algorithm for a given set of data.
		1.1.5	understand how to identify and correct errors in algorithms	Same	1.1.5	Understand how to identify and correct errors in algorithms, including using trace tables.
		1.1.6	understand how to code an algorithm in a high-level language	Same	1.1.6	Understand how to code an algorithm in a high-level language.
		1.1.7	understand how the choice of algorithm is influenced by the data structures and data values that need to be manipulated	Same	1.1.7	Understand how the choice of algorithm is influenced by the data structures and data values that need to be manipulated.
		1.1.8	understand how standard algorithms (bubble sort, merge sort, linear search, binary search) work	Same	1.1.8	Understand how standard algorithms work (bubble sort, merge sort, linear search, binary search).
		1.1.9	be able to evaluate the fitness for purpose of algorithms in meeting specified requirements efficiently using logical reasoning and test data	Same	1.1.9	Be able to evaluate the fitness for purpose of algorithms in meeting specified requirements efficiently, using logical reasoning and test data.
	Decomposition and abstraction	1.2.1	be able to analyse a problem, investigate requirements (inputs, outputs, processing, initialisation) and design solutions	Same	1.2.1	Be able to analyse a problem, investigate requirements (inputs, outputs, processing, initialisation) and design solutions.
		1.2.2	be able to decompose a problem into smaller sub-problems	Same	1.2.2	Be able to decompose a problem into smaller sub-problems.
		1.2.3	understand how abstraction can be used effectively to model aspects of the real world	Same	1.2.3	Understand how abstraction can be used effectively to model aspects of the real world.
		1.2.4	be able to program abstractions of real-world examples	Same	1.2.4	Be able to program abstractions of real-world examples.

Topic	Sub Topic	GCSE Reference	GCSE Learning Objective	Level of difference	International GCSE Reference	International GCSE Learning Objective
Programming	Develop code	2.1.1	be able to write programs in a high-level programming language	Same	2.1.1	Be able to program abstractions of real-world examples.
		2.1.2	understand the benefit of producing programs that are easy to read and be able to use techniques (comments, descriptive names (variables, constants, subprograms), indentation) to improve readability and to explain how the code works	Same	2.1.2	Understand the benefit of producing programs that are easy to read and be able to use techniques (comments, descriptive names (variables, constants, subprograms), indentation) to improve readability and to explain how the code works.
		2.1.3	be able to differentiate between types of error in programs (logic, syntax, runtime)	Same	2.1.3	Be able to differentiate between types of error in programs (logic, syntax, runtime).
		2.1.4	be able to design and use test plans and test data (normal, boundary, erroneous)	Same	2.1.4	Be able to design and use test plans and test data (normal, boundary, erroneous).
		2.1.5	be able to interpret error messages and identify, locate and fix errors in a program	Same	2.1.5	Be able to interpret error messages and identify, locate and fix errors in a program.
		2.1.6	be able to determine what value a variable will hold at a given point in a program (trace table)	Same	2.1.6	Be able to determine what value a variable will hold at a given point in a program (trace table).
		2.1.7	be able to determine the strengths and weaknesses of a program and suggest improvements	Same	2.1.7	Be able to determine the strengths and weaknesses of a program and suggest improvements.
	Constructs	2.2.1	understand the structural components of a program (variable and type declarations, command sequences, selection, iteration, data structures, subprograms)	Same	2.2.1	Understand the structural components of a program (variable and type declarations, command sequences, selection, iteration, data structures, subprograms).
		2.2.2	be able to use sequencing, selection and iteration constructs in their programs	Same	2.2.2	Be able to use sequencing, selection and iteration constructs in their programs.
	Data types and structures	2.3.1	understand the need for, and understand how to use, data types (integer, real, Boolean, char)	Same	2.3.1	Understand the need for, and understand how to use, data types (integer, real, Boolean, char, string).
		2.3.2	understand the need for, and understand how to use, data structures (records, one-dimensional arrays, two-dimensional arrays)	Same	2.3.2	Understand the need for, and understand how to use, data structures (records, one-dimensional arrays, two-dimensional arrays).
		2.3.3	understand the need for, and how to manipulate, strings	Same	2.3.3	Understand the need for, and how to manipulate, strings.
		2.3.4	understand the need for, and how to use, variables and constants	Same	2.3.4	Understand the need for, and how to use, variables and constants.
		2.3.5	understand the need for, and how to use, global and local variables when implementing subprograms	Same	2.3.5	Understand the need for, and how to use, global and local variables when implementing subprograms.
	Input / Output	2.4.1	understand how to write code that accepts and responds appropriately to user input	Same	2.4.1	Understand how to write code that accepts and responds appropriately to user input.
		2.4.2	understand the need for, and how to implement, validation	Same	2.4.2	Understand the need for, and how to implement, validation
		2.4.3	be able to write code that reads/writes from/to a text file	Same	2.4.3	Be able to write code that reads/writes from/to a text file.
	Operators	2.5.1	understand the purpose of, and how to use, arithmetic operators (add, subtract, divide, multiply, modulus, integer division)	Same	2.5.1	Understand the purpose of, and how to use, arithmetic operators (add, subtract, divide, multiply, modulus, integer division).
		2.5.2	understand how to use relational operators (equal to, less than, greater than, not equal to, less than or equal to, greater than or equal to)	Same	2.5.2	Understand the purpose of, and how to use, relational operators (equal to, less than, greater than, not equal to, less than or equal to, greater than or equal to).
		2.5.3	understand the purpose of, and how to use, logic operators (AND, OR, NOT)	Same	2.5.3	Understand the purpose of, and how to use, logic operators (AND, OR, NOT).
	Subprograms	2.6.1	understand the benefits of using subprograms and be able to write code that uses user-written and pre-existing (built-in, library) subprograms	Same	2.6.1	understand the benefits of using subprograms and be able to write code that uses user-written and pre-existing (builtin, library) subprograms
		2.6.2	understand the concept of passing data into and out of subprograms (procedures, functions)	Same	2.6.2	understand the concept of passing data into and out of subprograms (procedures, functions)
		2.6.3	be able to create subprograms that use parameters	Same	2.6.3	be able to create subprograms that use parameters

Topic	Sub Topic	GCSE Reference	GCSE Learning Objective	Level of difference	International GCSE Reference	International GCSE Learning Objective
Data	Binary	3.1.1	understand that computers use binary to represent data (numbers, text, sound, graphics) and program instructions	Same	3.1.1	Understand that computers use binary to represent data (numbers, text, sound, graphics) and program instructions.
		3.1.2	understand how computers represent and manipulate numbers (unsigned integers, signed integers (sign and magnitude, two's complement))	Same	3.1.2	Understand how computers represent and manipulate numbers (unsigned integers, signed integers (sign and magnitude, two's complement)).
		3.1.3	be able to convert between binary and denary whole numbers (0–255)	Same	3.1.3	Be able to convert between binary and denary whole numbers (0–255).
		3.1.4	understand how to perform binary arithmetic (add, shifts (logical and arithmetic)) and understand the concept of overflow	Same	3.1.4	Understand how to perform binary arithmetic (add, shifts (logical and arithmetic)) and understand the concept of overflow.
		3.1.5	understand why hexadecimal notation is used and be able to convert between hexadecimal and binary	Same	3.1.5	Understand why hexadecimal notation is used and be able to convert between hexadecimal and binary
	Data representation	3.2.1	understand how computers encode characters using ASCII	Similar	3.2.1	Understand how computers encode characters using ASCII and Unicode.
		3.2.2	understand how bitmap images are represented in binary (pixels, resolution, colour depth)	Same	3.2.2	Understand how bitmap images are represented in binary (pixels, resolution, colour depth).
		3.2.3	understand how sound, an analogue signal, is represented in binary	Same	3.2.3	Understand how sound, an analogue signal, is represented in binary.
		3.2.4	understand the limitations of binary representation of data (sampling frequency, resolution) when constrained by the number of available bits	Same	3.2.4	Understand the limitations of binary representation of data (sampling frequency, resolution) when constructed by the number of available bits.
	Data storage and compression	3.3.1	understand how to convert between the terms 'bit, nibble, byte, kilobyte (KB), megabyte (MB), gigabyte (GB), terabyte (TB)'	Different	3.3.1	Understand how to use and convert between binary and denary multiples (as defined by the International Electrotechnical Commission (IEC)): bit, nibble, byte, kibibyte (KiB) 2^10, mebibyte (MiB) 2^20, gibibyte (GiB) 2^30, tebibyte (TiB) 2^40, kilobyte (kB), 10^3, megabyte (MB) 10^6, gigabyte (GB) 10^9, terabyte (TB) 10^12
		3.3.2	understand the need for data compression and methods of compressing data (lossless, lossy) and that JPEG and MP3 are examples of lossy algorithms	Same	3.3.2	Understand the need for data compression and methods of compressing data (lossless, lossy), and that JPEG and MP3 are examples of lossy algorithms.
		3.3.3	understand how a lossless, run-length encoding (RLE) algorithm works	Same	3.3.3	Understand how a lossless, run-length encoding (RLE) algorithm works
		3.3.4	understand that file storage is measured in bytes and be able to calculate file sizes	Same	3.3.4	Understand that file storage is measured in bytes and be able to calculate file sizes.
	Encryption	3.4.1	understand the need for data encryption	Same	3.4.1	understand the need for data encryption
		3.4.2	understand how a Caesar cipher algorithm works	Different	3.4.2	Understand how encryption algorithms work (Pigpen cipher, Caesar cipher, Vigenére cipher, Rail Fence cipher).
	Databases	3.5.1	understand the characteristics of structured and unstructured data	Different		
		3.5.2	understand that data can be decomposed, organised and managed in a structured database (tables, records, fields, relationships, keys)	Different		

Topic	Sub Topic	GCSE Reference	GCSE Learning Objective	Level of difference	International GCSE Reference	International GCSE Learning Objective
Computers	Machines and computational models	4.1.1	understand the input-process-output model	Same	4.1.1	Understand the input-process-output model.
				Different	4.1.2	Understand that there is a range of computational models (sequential, parallel, multi-agent).
	Hardware	4.2.1	understand the function of the hardware components of a computer system (CPU, main memory, secondary storage) and how they work together	Similar	4.2.1	Understand the function of the hardware components of a computer system (central processing unit (CPU), main memory, secondary storage, input and output devices) and how they work together.
		4.2.2	understand the function of different types of main memory (RAM, ROM, cache)	Similar	4.2.2	Understand the function of different types of memory (random-access memory (RAM), read-only memory (ROM), cache, virtual memory).
		4.2.3	understand the concept of a stored program and the role of components of the CPU (control unit (CU), arithmetic/logic unit (ALU), registers, clock, address bus, data bus) in the fetch-decode-execute cycle (the Von Neumann model)	Same	4.2.3	Understand the concept of a stored program and the role of components of the CPU (control unit (CU), arithmetic/logic unit (ALU), registers, clock, address bus, data bus, control bus) in the fetch-decodeexecute cycle (the Von Neumann model).
				Different	4.2.4	Understand factors that affect the performance of the CPU (clock speed, number of processor cores, size of cache, type of cache).
		4.2.4	understand how data is stored on physical devices (magnetic, optical, solid state)	Same	4.2.5	Understand how data is stored on physical devices (magnetic, optical, solid state).
		4.2.5	understand the concept of storing data in the 'cloud' and other contemporary secondary storage	Same	4.2.6	Understand the concept of storing data in the 'cloud' and other contemporary secondary storage.
		4.2.6	understand the need for embedded systems and their functions	Same	4.2.7	Understand the need for embedded systems and their functions.
	Logic	4.3.1	be able to construct truth tables for a given logic statement (AND, OR, NOT)	Same	4.3.1	be able to construct truth tables for a given logic statement (AND, OR, NOT)
		4.3.2	be able to produce logic statements for a given problem	Same	4.3.2	be able to produce logic statements for a given problem
	Software	4.4.1	know what an operating system is and how it manages files, processes, hardware and the user interface	Same	4.4.1	Know what an operating system is and how it manages files, processes, hardware and the user interface.
		4.4.2	understand the purpose and functions of utility software (managing, repairing and converting files; compression; defragmentation; backing up; anti-virus, anti-spyware)	Same	4.4.2	Understand the purpose and functions of utility software (managing, repairing and converting files; compression; defragmentation; backing up; anti-malware (antivirus, anti-spyware)).
		4.4.3	understand how software can be used to simulate and model aspects of the real world	Same	4.4.3	Understand how software can be used to simulate and model aspects of the real world.
	Programming languages	4.5.1	understand what is meant by high-level and low-level programming languages and understand their suitability for a particular task	Same	4.5.1	Understand what is meant by high-level and low-level programming languages and understand their suitability for a particular task.
		4.5.2	understand what is meant by a compiler and an interpreter and know the advantages and disadvantages of each.	Similar	4.5.2	Understand what is meant by an assembler, a compiler and an interpreter when translating programming languages and know the advantages and disadvantages of each.

Topic	Sub Topic	GCSE Reference	GCSE Learning Objective	Level of difference	International GCSE Reference	International GCSE Learning Objective
Communication and the Internet	Networks	5.1.1	understand why computers are connected in a network	Same	5.1.1	understand why computers are connected in a network
		5.1.2	understand the different types of networks (LAN, WAN) and usage models (client-server, peer-to-peer)	Similar	5.1.2	Understand the different types of networks (local area network (LAN), wide area network (WAN), personal area network (PAN)) and usage models (client-server, peer-to-peer).
		5.1.3	understand wired and wireless connectivity	Same	5.1.3	understand wired and wireless connectivity
		5.1.4	understand that network data speeds are measured in bits per second (Mbps, Gbps)	Same	5.1.4	understand that network data speeds are measured in bits per second (Mbps, Gbps)
		5.1.5	understand the role of and need for network protocols (Ethernet, Wi-Fi, TCP/IP, HTTP. HTTPS, FTP, email (POP3, SMTP, IMAP))	Same	5.1.5	understand the role of and need for network protocols (Ethernet, Wi-Fi, TCP/IP, HTTP. HTTPS, FTP, email (POP3, SMTP, IMAP))
		5.1.6	understand that data can be transmitted in packets using layered protocol stacks (TCP/IP)	Similar	5.1.6	Understand that data can be transmitted in packets using layered protocol stacks and the 4-layer TCP/IP model (application, transport, network, data link).
		5.1.7	understand characteristics of network topologies (bus, ring, star, mesh)	Same	5.1.7	understand characteristics of network topologies (bus, ring, star, mesh)
				Same	5.1.8	Understand the different mobile communication standards (3G, 4G and subsequent generations).
	Network security	5.2.1	understand the importance of network security and be able to use appropriate validation and authentication techniques (access control, physical security and firewalls)	Same	5.2.1	understand the importance of network security and be able to use appropriate validation and authentication techniques (access control, physical security and firewalls)
		5.2.2	understand security issues associated with the 'cloud' and other contemporary storage	Same	5.2.2	understand security issues associated with the 'cloud' and other contemporary storage
		5.2.3	understand different forms of cyberattack (based on technical weaknesses and behaviour) including social engineering (phishing, shoulder surfing), unpatched software, USB devices, digital devices and eavesdropping	Similar	5.2.3	Understand different forms of cyber attack (based on technical weaknesses and behaviour), including social engineering (phishing, shoulder surfing, pharming), unpatched software, USB devices, digital devices and eavesdropping.
		5.2.4	understand methods of identifying vulnerabilities including penetration testing, ethical hacking, commercial analysis tools and review of network and user policies	Same	5.2.4	understand methods of identifying vulnerabilities including penetration testing, ethical hacking, commercial analysis tools and review of network and user policies
		5.2.5	understand how to protect software systems from cyber attacks, including considerations at the design stage, audit trails, securing operating systems, code reviews to remove code vulnerabilities in programming languages and bad programming practices, modular testing and effective network security provision	Similar	5.2.5	Understand how to protect software systems from cyber attacks, including considerations at the software (application) design stage, audit trails, securing operating systems, secure coding, code reviews to remove code vulnerabilities in programming languages and bad programming practices, modular testing and effective network security provision.
	The internet and the world wide web	5.3.1	understand what is meant by the internet and how the internet is structured (IP addressing, routers)	Similar	5.3.1	Understand what is meant by the internet and how the internet is structured (IP addressing, domain name service (DNS)).
		5.3.2	understand what is meant by the world wide web (WWW) and components of the WWW (web server URLs, ISP, HTTP, HTTPS, HTML)	Same	5.3.2	understand what is meant by the world wide web (WWW) and components of the WWW (web server URLs, ISP, HTTP, HTTPS, HTML)
				Different	5.3.3	Understand the need for IP addressing standards and the formats of IPv4 and IPv6.
				Different	5.3.4	Understand the role of components used to access the internet (modem, router, switch, wireless access point (WAP)) and how these are combined.

Topic	Sub Topic	GCSE Reference	GCSE Learning Objective	Level of difference	International GCSE Reference	International GCSE Learning Objective
The bigger picture	Emerging trends, issues and impact	6.1.1	understand the environmental impact of technology (health, energy use, resources) on society	Same	6.1.1	understand the environmental impact of technology (health, energy use, resources) on society
		6.1.2	understand the ethical impact of using technology (privacy, inclusion, professionalism) on society	Same	6.1.2	understand the ethical impact of using technology (privacy, inclusion, professionalism) on society
		6.1.3	understand the legal impact of using technology (intellectual property, patents, licensing, open source and proprietary software, cyber-security) on society	Same	6.1.3	Understand the legal impact of using technology (intellectual property, patents, licensing and cyber-security).
				Different	6.1.4	Be aware of current and emerging trends in computing technology (quantum computing, DNA computing, artificial intelligence (AI), nanotechnology).