The impact of Edexcel's change from 1CP1 to 1CP2 specifications

VERSION 4

Brought to you by www.gcsecs.org, the home of:

- Model answers to Programming Projects (former NEA) for teachers only
- The Ultimate GCSE Computer Science Textbook
- The Ultimate GCSE Computer Science Animated Presentations

The above textbook and presentations will be updated to reflect the new 1CP2 specification in time for September 2020.

Version Control

Version 1:

- Based on Edexcel's draft specification

Version 2 Updates:

- Based on Edexcel's Issue 1 specification 1CP2
- New specification codes added
- Assessment Reference Language updated to Programming Language Subset
- Learning objectives updated from draft specification
- Comments updated to reflect updated learning objectives

Version 3 Updates:

- Some corrections following feedback from Edexcel
- Updated to reflect information in Edexcel's Getting Started Guide

Feedback on this document is welcome to paul@gcsecs.org

Contents

The impact of Edexcel's change from 1CP1 to 1CP2 specifications
VERSION 3

Brought to you by www.gcsecs.org, the home of:

- Model answers to Programming Projects (former NEA) for teachers only
- The Ultimate GCSE Computer Science Textbook
- The Ultimate GCSE Computer Science Animated Presentations

The above textbook and presentations will be updated to reflect the new 1CP2 specification in time for September 2020.

When do I teach each specification?

1CP1 specification is for examination in summer 2020 and summer 2021:

- Year 10 who started in September 2019 (unless taking exams at end of year 10)
- Year 11 who started in Year 10 in September 2018

1CP2 specification is for examination from summer 2022:

- Year 9 starting in September 2019 (unless taking exams at end of year 10)
- Year 10 starting in September 2020

What's happening with the Programming Project (former NEA)?

When the original Edexcel specification was launched, it included an element of 20% assessment by Non-Examined Assessment (NEA). This was cancelled during the first year due to concerns by Ofqual that the authenticity of the assessment couldn't be guaranteed. It was replaced by a Programming Project (PP) which was not assessed, but centres had to show they had offered students the opportunity to spend 20 timetabled classroom hours on the PP which was a board set task by Edexcel. The two examinations were changed from 40% each to 50% each. Centres were required to submit a sample of student work to Edexcel for verification that students had completed the project. Centres also had to complete an authentication form and timesheet confirming that all students had been given the opportunity to spend 20 timetabled hours on the PP.

With the new Edexcel specification, the PP is being replaced by a Practical Programming Statement (PPS) that must confirm students have had the chance to complete practical programming experience throughout the course that cover design, write, test and refine. The key changes appear to be:

- No analysis
- No evaluation
- No set format except that students must have the opportunity to design, write, test and refine
- No specific time requirement
- No specific scope for the size of the project
- Assessment of programming skills will be in exam Paper 2.
- Centres will need to sign a PPS to confirm all students had an opportunity to undertake a project

As with the PP, any high-level language can be used, but it would be advisable to use Python 3 as this is what the exam, especially Paper 2, will be based on.

It's still an important part of the course because students will be assessed on their ability to design, write, test and refine in Paper 2 which is worth 50% of the whole qualification. Paper 2 will be an on-screen exam using Python 3 in an IDE of the school's choice. Mini projects will be available from www.gcsecs.org to support the programming element of the specification.

What's Programming Language Subset (PLS)?
OCR, AQA and Edexcel have all provided a version of their own pseudo-code that is used when writing exam questions to ensure consistency in the exams and that students are able to understand the command set. Pseudo-code wasn't really the right name for this as pseudo-code doesn't have any formal structure. Edexcel have therefore changed this to be a subset of Python called a Programming Language Subset (PLS).

The Programming Language Subset, as detailed in the specification, is the language that will be used when presenting code in in exam questions so that all students are able to interpret it, no matter what programming language they have used at school.

Edexcel's PLS is a subset of Python. It will be used if code is presented to students in an exam, particularly for paper 2. It will also be used for any sample code that is provided as part of the practical programming exam. All programming questions set by Edexcel in the exam should be able to be completed using just the PLS, but this does NOT stop students from using their own experience of programming with Python to find a different solution.

Changes to the draft
There was an opportunity for teachers to provide feedback directly to Edexcel after their first draft of the specification. The detailed feedback I gave to Edexcel was very welcome and they made many changes as a result of this. Following concerns that their specification was too vague, they have also produced a Getting Started Guide and made an early copy available to me so I could feedback on it and they have incorporated almost all of my feedback. Edexcel have shown that they are a responsive awarding body and willing to listen.

Changes from pseudo-code in 1CP1 specification to Programming Language Subset in 1CP2 specification
As the Programming Language Subset (PLS) uses Python, there is very little resemblance to the 1CP1 specification's pseudo-code. A comparison of each element of code therefore seems irrelevant. These are some highlights worth noting:

- <xxxx> is used to denote an expression or value, e.g. print(<var_name>)
- Constants will always be in UPPER CASE
- Arrays are included as a structured data type but you'll also need to include lists
- Block segments are indented and so there are no END IF, END WHILE etc but it would be good practice for students to use comments such as # END IF
- Additional code students need to understand includes:
  - Reading and writing files
  - ASCII conversion (see feedback above)
  - List methods
  - Random integers and floats
  - String methods
  - Formatting of strings
  - Math module functions
  - Sleep from the Time module
  - Turtle module functions

Have any of the exams changed?
There are still two papers worth 50% each:
1. Paper 1: Principles of Computer Science
2. Paper 2: Application of Computational Thinking

Both papers are now 75 marks each and not 80 marks each.

Paper 1 changes include:
- Paper 1 is now 1 hour and 30 minutes (used to be 1 hour and 40 minutes)
- There will be 5 compulsory questions focused on each of topics 1-5 in the subject content

Paper 2 changes include:
- it will now be an on-screen examination where students will use Python 3 within an IDE of the centre's choice
- pre-existing code files will be made available for students to refine, test and correct
- all questions will use the Programming Language Subset
- students will be using Python 3 and so syntax will be important in answers
- there will be 6 compulsory questions
- it will assess topic 6 from the subject content

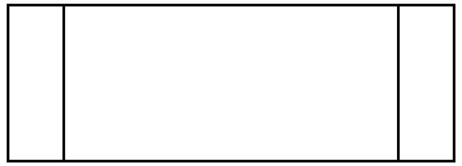
Other key changes are:

- Topic numbers have changed

What's changed from the original content?

New areas to teach include:

- Iteration over every item in data structures
- Efficiency of algorithms in terms of number of compares, number of passes, use of memory
- Determine maximum number of states represented by a binary bit pattern of a given length
- Convert 8 bit two's complement binary to decimal and vice versa
- Kibibyte, mebibyte, gibibyte and tebibyte
- Be able to construct expressions to calculate file sizes and data capacity requirements
- Impact of wired and wireless connectivity on performance
- Construct expressions involving file size, transmission rate and time
- Ethical and legal issues associated with artificial intelligence, machine learning and robotics
- Malware
- Methods of protecting digital systems
- Convert algorithms to programs and vice versa
- Single entry/exit points from code blocks
- Read/write to comma separated value text files (for the purpose of being tested in the exam)
- New flowchart symbol for "pre-defined subprogram":



Areas you no longer need to teach include:

- Investigating requirements
- Test plans and test data
- Interpreting error messages
- Sign and magnitude version of signed integers
- JPG and MP3 examples of lossy algorithms
- Run-length encoding
- Caesar cipher
- Structured and unstructured data
- Databases
- Input-process-output model
- Input devices
- ROM and cache
- Storing date in the ‘cloud’
- Produce logic statements
- Simulation and modelling
- Assemblers
- Client-server and peer-to-peer
- Ring topology
- Commercial analysis tools and review of network and user policies
- world wide web and its components

The changes are detailed in two tables below.

The first table looks at the changes from the perspective of the new 1CP2 specification so you can see the new structure and how it relates to the old 1CP1 specification you have already been teaching.

The second looks at the changes from the perspective of the old 1CP1 specification so you can adapt your teaching based on what you are already doing ready for the 1CP2 specification.

The spec codes refer to the numbering system used in each specification.

Comments in the "What difference does it make" draw upon information in both the specification and Edexcel's Getting Started Guide. It is strongly recommended that the Getting Started Guide (GSG) is looked at in parallel with these tables for further clarification of each specification point.

Green text means that something new has been added.

Red text means that something has been removed.

Orange text is used to highlight a change that isn't a clear addition or removal but some sort of other change, for example a clarification or change of terminology.

Changes in new 1CP2 specification order

New Topic 1: Computational thinking

1CP2		1CP1		What difference does it make?		
Spec	Learning Objective	Spec	Learning Objective	Impact Summary	What you need to do	Qualified opinion
1.1.1	understand the benefit of using decomposition and abstraction to model aspects of the real world and analyse, understand and solve problems	1.2.1	be able to analyse a problem, ~~investigate requirements (inputs, outputs, processing, initialisation) and design solutions~~	Low Removed	No need to cover investigating requirements. Decomposition and abstraction are also covered in 6.1.1 but 1.1.1 is specifically about understanding the benefits which will be tested in Paper 1.	Decomposition is about decomposing into smaller problems so that's probably why the text has been removed but it still needs teaching.
		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			
		1.2.4	be able to program abstractions of real-world examples			
1.1.2	understand the benefits of using 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	None	Teach this with topic 6 so that it includes writing subprograms. The GSG states students should know the difference between functions and procedures.	This is the theoretical element of the benefits to be tested in Paper 1. Using subprograms comes in Paper 2.
1.2.1	be able to follow and write algorithms (flowcharts, pseudocode, program code) that use sequence, selection, repetition (count-controlled, condition-controlled) and iteration (over every item in a data structure), and input, processing and output to solve problems	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)	High	Instead of teaching Edexcel's old pseudocode you need to use Python, focusing on their PLS. Informal pseudocode can still be used to answer algorithm questions in paper 1. Loops must include count-controlled (FOR) and pre-conditioned (WHILE) and post-conditioned (UNTIL). Expect loops to iterate through data structures. Although there's no need to cover input-process-output model, it does come in handy.	There is an error here in that iteration isn't just about data structures but also includes loops. With the PLS being Python, it's easier to teach just the Python language, but there is a wider scope now that needs covering. It will be interesting to see how they test post-conditioned loops in the exam when these can't be used in Python.
		4.1.1	~~understand the input-process-output model~~			
1.2.2	understand the need for and be able to follow and write algorithms that use variables and constants and one- and two-dimensional data structures (strings, records, arrays)	2.3.2	understand the need for, and understand how to use, data structures (records, one-dimensional arrays, two-dimensional arrays)	None	Although Python uses lists and not arrays, the principals of arrays and records as static data structures still needs to be covered including that items in an array must be the same data type.	Easiest to teach this alongside topic 6
		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			

1CP2		1CP1		What difference does it make?		
Spec	**Learning Objective**	**Spec**	**Learning Objective**	**Impact Summary**	**What you need to do**	**Qualified opinion**
1.2.3	understand the need for and be able to follow and write algorithms that use arithmetic operators (addition, subtraction, division, multiplication, modulus, integer division, exponentiation), relational operators (equal to, less than, greater than, not equal to, less than or equal to, greater than or equal to) and logical operators (AND, OR, NOT)	2.5.1	understand the purpose of, and how to use, arithmetic operators (add, subtract, divide, multiply, modulus, integer division)	Low Added	Include exponentiation in your teaching.	Easiest to teach this alongside topic 6
		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)	None		
		2.5.3	understand the purpose of, and how to use, logic operators (AND, OR, NOT)	None		
1.2.4	be able to determine the correct output of an algorithm for a given set of data and use a trace table to determine what value a variable will hold at a given point in an algorithm	1.1.4	~~Understand how~~ to determine the correct output of an algorithm for a given set of data	Low Clarify	Nothing	This is just a clarification that students need to be able to use a trace table rather than just understand how to use one.
		2.1.6	be able to determine what value a variable will hold at a given point in a program (trace table)			
1.2.5	understand types of errors that can occur in programs (syntax, logic, runtime) and be able to identify and correct in algorithms	1.1.5	Understand how to identify and correct errors in algorithms	Low Clarify	Be aware that errors in algorithms are always logic errors, but errors in programming code can be logic, syntax or runtime.	This is just a clarification that students need to be able to identify and correct errors rather than just understand how to do it. Probably best to teach 1.2.5 with 6.1.5.
		2.1.3	be able to differentiate between types of error in programs (logic, syntax, runtime)			
1.2.6	understand how standard algorithms (bubble sort, merge sort, linear search, binary search) work	1.1.8	understand how standard algorithms (bubble sort, merge sort, linear search, binary search) work	None	Students only need to be able to code the linear search. Cover the comparison of the different sorts and searches.	
1.2.7	be able to use logical reasoning and test data to evaluate an algorithm's fitness for purpose and efficiency (number of compares, number of passes through a loop, use of memory)	1.1.9	be able to evaluate the fitness for purpose of algorithms ~~in meeting specified requirements~~ efficiently using logical reasoning and test data	Low Clarified	No need to evaluate an algorithm against its requirements. Cover in teaching how to evaluate efficiency in terms of the number of compares, passes and use of memory. This should be also be done for search and sort methods including best- and worse-case scenarios for searches and being aware that a merge sort is an 'out of place sort' and so requires more memory than a bubble sort	This has been clarified as to what efficiency means. It's confusing that it appears in both 1.2.7 and 6.1.6 of 1CP2 spec.
		2.1.7	be able to determine the strengths and weaknesses of a program and suggest improvements			
1.3.1	be able to apply logical operators (AND, OR, NOT) in appropriate truth tables with up to three inputs to solve problems	4.3.1	be able to construct truth tables for a given logic statement (AND, OR, NOT)	Low Removed	No need to teach logic statements. Ensure you are giving students plenty of practice at using truth tables for single and multiple logical operators. No need to be able to draw logic diagrams but should be able to interpret them.	Although they don't need to be taught, it's useful to understand how logic statements work in a truth table in order to be able to apply them to different situations.

New Topic 2: Data

1CP2		1CP1		What difference does it make?		
Spec	Learning Objective	Spec	Learning Objective	Impact Summary	What you need to do	Qualified opinion
2.1.1	understand that computers use binary to represent data (numbers, text, sound, graphics) and program instructions and be able to determine the maximum number of states that can be represented by a binary pattern of a given length	3.1.1	understand that computers use binary to represent data (numbers, text, sound, graphics) and program instructions	Medium Added	Teach that 2^n gives the number of unique values that n bits can represent	This is straightforward
2.1.2	understand how computers represent and manipulate unsigned integers and two's complement signed integers	3.1.2	understand how computers represent and manipulate ~~numbers~~ (unsigned integers, signed integers (~~sign and magnitude~~, two's complement))	Medium Removed	Remove sign and magnitude method of signed integers from your teaching.	This makes a lot of sense as the two methods were confusing when taught alongside each other at this level.
2.1.3	be able to convert between denary and 8-bit binary numbers (0 – 255, -127 – 128)	3.1.3	be able to convert between binary and denary whole numbers (0–255)	Low Clarified	If not already doing it, ensure students are taught how to convert 8-bit signed binary numbers.	This clarifies that negative binary numbers need to be converted too.
2.1.4	be able to add together two positive binary patterns and apply logical and arithmetic binary shifts	3.1.4	understand how to perform binary arithmetic (add, shifts (logical and arithmetic)) and understand the concept of overflow	Low Added	Cover why a right binary shift results in a lack of precision (DIV rather than ÷)	It's not clear whether students need to understand the additional issues with 2's complement shifts.
2.1.5	understand the concept of overflow in relation to the number of bits available to store a value	3.1.4	understand how to perform binary arithmetic (add, shifts (logical and arithmetic)) and understand the concept of overflow	Low Clarified	Nothing	This just expands on what overflow is.
2.1.6	understand why hexadecimal notation is used and be able to convert between hexadecimal and binary	3.1.5	understand why hexadecimal notation is used and be able to convert between hexadecimal and binary	None		
2.2.1	understand how computers encode characters using 7-bit ASCII	3.2.1	understand how computers encode characters using ASCII	Low Clarified	No need to teach about 8-bit extended ASCII and Unicode in depth but students need to be aware that these provide larger character sets.	
2.2.2	understand how bitmap images are represented in binary (pixels, resolution, colour depth)	3.2.2	understand how bitmap images are represented in binary (pixels, resolution, colour depth)	High Added	The GSG requires an understanding of resolution (ppi) and calculating the physical size of an image. The GSG wants students to be aware that higher resolutions result in higher file sizes. As well as constructing an expression for the file size, students need to be able to calculate a missing value in the equation, e.g. colour depth. They also need to be able to convert binary data into a 2 colour (colour depth of 1 bit) image	Resolution vs image size is a difficult concept so it's a shame it's been included, especially as resolution isn't always directly associated with file size as it's related to output rather than storage.

1CP2		1CP1		What difference does it make?		
Spec	Learning Objective	Spec	Learning Objective	Impact Summary	What you need to do	Qualified opinion
2.2.3	understand how analogue sound is represented in binary (amplitude, sample rate, bit depth, sample interval)	3.2.3	understand how sound, an analogue signal, is represented in binary	Low Added	Ensure you are teaching the terms amplitude, sample rate, bit depth and sample interval. As well as being able to construct an expression for the file size, students need to be able to calculate a missing value in the equation, e.g. bit depth.	Sample interval isn't actually used to represent sound in binary but it's been included in the specification so needs teaching.
2.2.4	understand the limitations of binary representation of data (sampling frequency, resolution) when constrained by the number of available bits	3.2.4	understand the limitations of binary representation of data (sampling frequency, resolution) when constrained by the number of available bits	Low Clarified	It's possible that the bit in brackets is being removed by Edexcel from the specification.	There are some errors in terminology here being inconsistent with 2.2.3 which uses correct terminology. It's also confusing because the bit depth (resolution) is the number of bits per sample which is what the constraint is.
2.3.1	understand that data storage is measured in binary multiples (bit, nibble, byte, kibibyte, mebibyte, gibibyte, tebibyte) and be able to construct expressions to calculate file sizes and data capacity requirements	3.3.1	understand how to convert between the terms 'bit, nibble, byte, kilobyte (KB), megabyte (MB), gigabyte (GB), terabyte (TB)'	Low Clarified Added	Ensure you teach the difference between kilobytes (1000) and kibibytes (1024). Cover the use of expressions to show how a file size would be calculated for text files, image files and sound files. 1024 must be used, not 1000.	The use of kibibytes etc will be because Edexcel were wrongly using 1024 in the exam for kilobytes which are actually 1000. However, if they had stuck to kilobytes etc then calculations could be made instead of expressions. Although expressions weren't on the 2012 specification, they were expected in the exams.
		3.3.4	understand that file storage is measured in bytes and be able to calculate file sizes			
2.3.2	understand the need for data compression and methods of compressing data (lossless, lossy)	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~~	Medium Removed	No need to teach examples of JPEG, MPE and RLE but examples are usually helpful. The GSG says pros and cons of lossy vs lossless should be covered.	This means specific questions in the exam won't be asked about these examples, but higher ability students can still be stretched by using examples.
		3.3.3	~~understand how a lossless, run-length encoding (RLE) algorithm works~~			

1CP2		1CP1		What difference does it make?		
Spec	Learning Objective	Spec	Learning Objective	Impact Summary	What you need to do	Qualified opinion
3.1.1	understand the von Neumann stored program concept and the role of main memory (RAM), CPU (control unit, arithmetic logic unit, registers), clock, address bus, data bus, control bus in the fetch-decode-execute cycle	4.2.1	understand the function of the hardware components of a computer system (CPU, main memory, secondary storage~~, input and output devices) and how they work together~~	Low Removed	No need to teach input and output devices No need to understand how devices work together No need to cover ROM and cache Need to know the purpose of registers but no need to name them or describe their function.	Input and output devices are very basic, but they were good for Grade 1 and 2 students. ROM and cache were always good differentiators for higher ability students.
		4.2.2	understand the function of ~~different types~~ of main memory (RAM, ~~ROM, cache~~)			
		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-decode-execute cycle (the Von Neumann model)			
3.1.2	understand the role of secondary storage and the ways in which data is stored on devices (magnetic, optical, solid state)	4.2.4	understand how data is stored on physical devices (magnetic, optical, solid state)	Low Clarified	Ensure you teach about how data is stored physically on magnetic, optical and solid state devices Include in your teaching the role/purpose of secondary storage (i.e. non-volatile storage). No need to cover 'cloud' storage. Students need to justify why one type of storage may be more suitable for a given purpose.	Data stored in the cloud is still stored on physical devices so it makes sense to remove it. The role of secondary storage is important as it's non-volatile unlike RAM. "How" instead of "ways in which" clarifies that students need to understand how data is stored on each medium.
		4.2.5	understand the concept of storing data in the 'cloud' and other contemporary secondary storage			
3.1.3	understand the concept of an embedded system and what embedded systems are used for	4.2.6	understand the need for embedded systems and their functions	Low Added	GSG states students should understand the role of a microcontroller and be familiar with the IoT and be aware of privacy and security concerns associated with it.	
3.2.1	understand the purpose and functionality of an operating system (file management, process management, peripheral management, user management)	4.4.1	know what an operating system is and how it manages files, processes, hardware and the user ~~interface~~	Medium Added Removed	No need to cover the user interface as a function of an operating system. Ensure your teaching now also includes peripheral management and user management as functions of the operating system. Ensure your teaching includes the purpose of an operating system.	

1CP2		1CP1		What difference does it make?		
Spec	Learning Objective	Spec	Learning Objective	Impact Summary	What you need to do	Qualified opinion
3.2.2	understand the purpose and functionality of utility software (file repair, backup, data compression, disc defragmentation, anti-malware)	4.4.2	understand the purpose and functions of utility software (~~managing~~, repairing and ~~converting~~ files; compression; defragmentation; backing up; anti-virus, anti-spyware)	Medium Clarified Removed	No need to cover managing and converting of files. Now need to cover more methods of anti-malware which will include ransomware and keyloggers as suggested in 5.3.1 of 1CP2 spec.	A bit of cross-referencing finds the list of anti-malware in 5.3.1 of 1CP2 spec but it would have been useful here in the specification.
3.2.3	understand the importance of developing robust software and methods of identifying vulnerabilities (audit trails, code reviews)	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	Medium Removed	No need to cover "securing operating systems" or modular testing.	This has simplified this section and opened up the reasons (importance) for developing robust systems. Network security provision is covered elsewhere.
3.3.1	understand the characteristics and purposes of low-level and high-level 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~~	Low Clarified	Nothing as this would probably be taught anyway.	Although the part about understanding suitability for a particular task has been removed, it's difficult to teach the difference without giving examples of how they would be used.
3.3.2	understand how an interpreter differs from a compiler in the way it translates high-level code into machine code	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.~~	Low Removed	No need to teach about assemblers. Any difference, not just advantages and disadvantages, should be covered for compilers and interpreters.	This makes it much clearer.

New Topic 4: Networks

1CP2		1CP1		What difference does it make?		
Spec	**Learning Objective**	**Spec**	**Learning Objective**	**Impact Summary**	**What you need to do**	**Qualified opinion**
4.1.1	understand why computers are connected in a network	5.1.1	understand why computers are connected in a network	None		
4.1.2	understand different types of networks (LAN, WAN)	5.1.2	understand the different types of networks (LAN, WAN) ~~and usage models (client-server, peer-to-peer)~~	Medium Removed	No need to teach about usage models.	This is a fairly big section to remove and something that is usually easy to teach as students have experience of client-server at school and peer-to-peer at home.
4.1.3	understand how the internet is structured (IP addressing, routers)	5.3.1	understand ~~what is meant by the internet and~~ how the internet is structured (IP addressing, routers)	Low Removed	No need to know 'what is meant by the internet'. GSG states students need to know about packet switching and packet headers.	Although this has been removed, it would be impossible to teach this without an understanding of what the internet is.
4.1.4	understand how the characteristics of wired and wireless connectivity impact on performance (speed, range, latency, bandwidth)	5.1.3	understand wired and wireless connectivity	Medium Added	Cover how the characteristics of both wired and wireless networks impact on performance. Wired should include fibre and copper.	This is a sensible addition.
4.1.5	understand that network speeds are measured in bits per second (kilobit, megabit, gigabit) and be able to construct expressions involving file size, transmission rate and time	5.1.4	understand that network data speeds are measured in bits per second (Mbps, Gbps)	Low Clarified Added	Teach students how to create expressions for calculating transmission rates. Ensure this section is taught using base-10 units and is based on bits (not bytes) and 2.3.1 is taught using base 2 and is based on bytes.	Edexcel had already been asking students to create expressions in exams even though it wasn't in the specification. The use of kilobit, megabit and gigabit is clear, but it may be confusing now kilobyte etc has been replaced with kibibyte etc in 2.3.1 meaning that students may get confused between base 2 in 2.3.1 and base 10 in 4.1.5.
4.1.6	understand the role of and need for network protocols (Ethernet, Wi-Fi, TCP/IP, HTTP, HTTPS, FTP and email protocols (POP3, SMTP, IMAP))	5.1.5	understand the role of and need for network protocols (Ethernet, Wi-Fi, TCP/IP, HTTP. HTTPS, FTP, email (POP3, SMTP, IMAP))	None	No need to know individual protocols that make up the Ethernet and Wi-Fi 'families'.	The removal of the word network makes no difference.
4.1.7	understand how the 4-layer (application, transport, internet, link) TCP/IP model handles data transmission over a network	5.1.6	understand that data can be transmitted in packets using layered protocol stacks (TCP/IP)	Medium Clarified	Continue to teach the TCP/IP model including packet switching which is used at the transport layer. Ensure other layers are also covered and how data moves up and down the stack.	
4.1.8	understand characteristics of network topologies (bus, star, mesh)	5.1.7	understand characteristics of network topologies (bus, ~~ring~~, star, mesh)	Low Removed	No need to cover ring networks.	This is a welcome change as ring networks are very rare. Perhaps removing bus networks would also have been a good idea.

1CP2		1CP1		What difference does it make?		
Spec	Learning Objective	Spec	Learning Objective	Impact Summary	What you need to do	Qualified opinion
4.2.1	understand the importance of network security, ways of identifying network vulnerabilities (penetration testing, ethical hacking) and methods of protecting networks (access control, physical security, firewalls)	5.2.4	understand methods of identifying vulnerabilities including penetration testing, ethical hacking, ~~commercial analysis tools and review of network and user policies~~	Low Removed	No need to cover commercial analysis tools and review of network and user policies. Validation and authentication are still covered but in 6.4.3 and 6.4.4 so will only be tested in Paper 2. Both white and black box pen testing should be covered including why they are needed.	This clarifies this much better. The old examples in brackets of authentication techniques were confusing so calling them methods of protecting networks is much better.
		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)			

New Topic 5: Issues and impact

1CP2		1CP1		What difference does it make?		
Spec	Learning Objective	Spec	Learning Objective	Impact Summary	What you need to do	Qualified opinion
5.1.1	understand environmental issues associated with the use of digital devices (energy consumption, manufacture, replacement cycle, disposal)	6.1.1	understand the environmental impact of technology (~~health~~, energy use, resources) on society	High Removed Added	No longer need to cover health when teaching about environmental issues. Need to cover manufacture, replacement cycle and disposal for environmental issues including how to reduce the environmental impact.	The additions make sense. Although health has been removed, the disposal and replacement cycle (and thus recycling) does have an effect on health, particularly in 3rd world countries.
5.2.1	understand ethical and legal issues associated with the collection and use of personal data (privacy, ownership, consent, misuse, data protection)	6.1.2	understand the ethical impact of using technology (privacy, ~~inclusion, professionalism~~) on society	High Changed	No need to cover inclusion and professionalism. Focus is now on personal data and should include ownership, consent, misuse and data protection. The UK Data Protection Act 2018 should be covered including the principals, right to be forgotten, rights of data subjects and obligations of data users. Students should be aware of the Privacy and Electronic Communications Regulations (PECR) in relation to cookies for collecting personal data. Students should be aware of how the Computer Misuse Act helps to protect individuals' personal data by deterring hackers.	This is good because it narrows down the focus to issues to do with personal data rather than any sort of technology.
5.2.2	understand ethical and legal issues associated with the use of artificial intelligence, machine learning and robotics (accountability, safety, algorithmic bias, legal liability)			High Added	This is a new section that needs teaching, but it is focused rather than open ended. Robotics could include autonomous machines, surgical robots, driverless cars, automated weapons and assistive robotics.	This is good because it narrows down the focus to issues to do with AI, machine learning and robotics rather than any sort of technology.
5.2.3	understand methods of intellectual property protection for computer systems and software (copyright, patents, trademarks, licencing)	6.1.3	understand ~~the legal impact of using technology~~ (intellectual property, patents, licensing, open source and proprietary software, ~~cyber-security~~) on society	High Added	Cyber security is covered in 5.3.1 and 5.3.2 so no need to relate it to this section. The focus is now on intellectual property rather than legal impacts on a range of aspects. Open source and proprietary software should still be covered as methods of licensing. In addition, copyright, patents and trademarks need to be covered including the difference between them. There's no need to know about specific legislation but there should be an awareness it exists.	This is much more succinct although it would be useful if licensing was more clearly defined.

1CP2		1CP1		What difference does it make?		
Spec	**Learning Objective**	**Spec**	**Learning Objective**	**Impact Summary**	**What you need to do**	**Qualified opinion**
5.3.1	understand the threat to digital systems posed by malware (viruses, worms, Trojans, ransomware, key loggers) and how hackers exploit technical vulnerabilities (unpatched software, out-of-date anti-malware) and use social engineering to carry out cyberattacks	5.2.2	understand security issues associated with the 'cloud' and other contemporary storage	High Clarified	Cover the digital threats as listed. Prevention methods are now all in 5.2.2. Social engineering now covers pretexting, phishing, baiting and quid pro quo techniques. No longer need to cover threats from USB devices and eavesdropping.	This is much neater to have cyber threats all in one place.
		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~~			
5.3.2	understand methods of protecting digital systems and data (anti-malware, encryption, acceptable use policies, backup and recovery procedures)	3.4.1	~~understand the need for data encryption~~	High Clarified	Cover the protection methods as listed. No longer need to cover the Caesar cipher algorithm, but it's still worth teaching as it helps to explain encryption well. Backup and recovery procedures should include RAID, off-site storage and standby equipment/premises. It's not necessary to cover specific backup methods.	This is much neater to have security measures all in one place.
		3.4.2	~~understand how a Caesar cipher algorithm works~~			

1CP2		1CP1		What difference does it make?		
Spec	**Learning Objective**	**Spec**	**Learning Objective**	**Impact Summary**	**What you need to do**	**Qualified opinion**
6.1.1	be able to use decomposition and abstraction to analyse, understand and solve problems	1.2.2	be able to decompose a problem into smaller sub-problems	Low Clarified	Decomposition and abstraction are also covered in 1.1.1 but 6.1.1 is specifically about being able to use these techniques to solve programming problems in Paper 2.	
		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			
6.1.2	be able to read, write, analyse and refine programs written in a high-level programming language	1.1.3	Understand the ~~purpose~~ of a given algorithm and ~~how an algorithm works~~	Low Clarified	The main change here is the ability to refine existing programs which could be presented electronically as Python files in Paper 2.	This is neatly put all in one place.
		1.1.6	~~understand how to code~~ an algorithm in a high-level language			
		2.1.1	be able to write programs in a high-level programming language			
6.1.3	be able to convert algorithms (flowcharts, pseudocode) into programs and convert programs into algorithms		New to Edexcel 1CP2	High Addition	Converting algorithms into programs was a necessary part of the old programming project so this skill still needs teaching. The main addition here is that if students are presented with a program in Python then they need to be able to turn it into a flowchart or pseudocode.	The GSG states that students may be required to convert algorithms provided for them or ones students have created themselves. The latter is disappointing as it can lead to double jeopardy with the same mistake being marked wrong twice.
6.1.4	be able to use techniques (layout, indentation, comments, meaningful identifiers, white space) to make programs easier to read, understand and maintain	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	Low Clarified	Adapt teaching notes for "descriptive names" to be "meaningful identifiers". Include layout and white space in your teaching.	This is much clearer. Although students don't need to answer questions in an exam about the benefits, it's good for them to be aware of them anyway.
6.1.5	be able to identify, locate and correct program errors (logic, syntax, runtime)	2.1.3	be able to differentiate between types of error in programs (logic, ~~syntax~~, runtime)	Low Removed	Syntax errors will only be used in Paper 2, probably as part of correcting a program that has already been written. Students still need to be able to identify and correct logic and runtime errors for Paper 1 – see 1.2.5 Students won't be assessed on interpreting error messages, although this is an essential part of debugging so will remain an essential part of teaching.	This makes sense as syntax errors are difficult to spot when written down, but when attempting to run a program using an interpreter then debugging comes into play.
		2.1.5	be able to ~~interpret error messages and~~ identify, locate and fix errors in a program			
6.1.6	be able to use logical reasoning and test data to evaluate a program's fitness for purpose and efficiency (number of compares, number of passes through a loop, use of memory)	1.1.9	be able to evaluate the fitness for purpose of algorithms ~~in meeting specified requirements~~ efficiently using logical reasoning and test data	Low Clarified	See 1.2.7	This is a duplication of 1.2.7. It's more likely to be tested in Paper 1 but more likely to be taught in Topic 6. The GSG incorrectly references SAM Paper 2 Q4 as being related to this
		2.1.7	be able to determine the strengths and weaknesses of a program and suggest improvements			

1CP2		1CP1		What difference does it make?		
Spec	Learning Objective	Spec	Learning Objective	Impact Summary	What you need to do	Qualified opinion
						specification point when it is actually related to 6.1.5.
6.2.1	understand the function of and be able to identify the structural components of programs (constants, variables, initialisation and assignment statements, command sequences, selection, repetition, iteration, data structures, subprograms, parameters, input/output)	2.2.1	understand the structural components of a program (variable and ~~type declarations~~, command sequences, selection, iteration, data structures, subprograms)	Low Added Removed	Include constants, initialisation, and assignment statements in your teaching. There's no need to include type declaration which makes sense as Python is being used. Parameters are covered in 6.6.2.	This is very unclear as to the difference between repetition and iteration. In the GSG Edexcel are referring to repetition being condition-controlled and count-controlled loops and iteration only being related to iterating over items in an array. This is incorrect use of terminology as iteration covers all of the above.
6.2.2	be able to write programs that make appropriate use of sequencing, selection, repetition (count-controlled, condition controlled), iteration (over every item in a data structure) and single entry/exit points from code blocks and subprograms	2.2.2	be able to use sequencing, selection and iteration constructs in their programs	Low Confused Added	Although there is an error here, this is tested in a practical sense and so you can teach the term iteration correctly. The inclusion of single entry/exit points from code blocks and subprograms would suggest that Edexcel wouldn't accept use of BREAK in Python, but their GSG states BREAK is acceptable as long as the logic of the code is still clear.	The suggestion that count-controlled and condition-controlled loops are repetition and not iteration is wrong. The suggestion that iteration only applies to items in a data structure is wrong. BREAK statements are a common method used in Python to mimic post-condition-controlled loops (repeat . . . until).
6.3.1	be able to write programs that make appropriate use of primitive data types (integer, real, Boolean, char) and one- and two-dimensional structured data types (string, array, record)	2.3.1	understand the need for, and understand how to use, data types (integer, real, Boolean, char)	None	Nothing	The addition of the word primitive doesn't change what needs to be covered. Note that the GSG contains an error that suggests the list data structure when using mixed data types is storing records. This isn't correct as records have fixed lengths and lists do not. It is correct though that records can contain mixed data types.
		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			
6.3.2	be able to write programs that make appropriate use of variables and constants	2.3.4	understand the need for, and how to use, variables and constants	None	In the GSG, Edexcel expect students to use CAPITALS to declare constants, although there is no implementation of constants in Python.	
6.3.3	be able to write programs that manipulate strings (length, position, substrings, case conversion)	2.3.3	understand the need for, and how to manipulate, strings	Low Clarification	Ensure teaching of manipulation of strings includes length, position, substrings and case conversion. See the PLS.	This clarifies which string manipulation methods/functions are necessary.
6.4.1	be able to write programs that accept and respond appropriately to user input	2.4.1	understand how to write code that accepts and responds appropriately to user input	Low Added	The PLS requires students to be able to use the string.format() method for formatting output.	

1CP2		1CP1		What difference does it make?		
Spec	Learning Objective	Spec	Learning Objective	Impact Summary	What you need to do	Qualified opinion
6.4.2	be able to write programs that read from and write to comma separated value text files	2.4.3	be able to write code that reads/writes from/to a text file	Low Clarification	It was already necessary to teach CSV import and export for the Programming Project, but this means it can now be tested in Paper 2 so ensure students are able to recall how to do this.	
6.4.3	understand the need for and be able to write programs that implement validation (length check, presence check, range check, pattern check)	2.4.2	understand the need for, and how to implement, validation	Low Clarification	Ensure all the validation methods listed are covered. Other security measures are included in 5.3.2 and 6.4.4	It's clear which validation techniques students need to cover.
		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)			
6.4.4	understand the need for and be able to write programs that implement authentication (ID and password, lookup)	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)	Low Clarification	Ensure that you are including ID, password and lookup of ID/password in a list in your teaching for implementation methods using Python. Differentiate as lower ability students will find lookup difficult.	This has been made much clearer and the incorrect examples of authentication techniques have been removed.
6.5.1	be able to write programs that use arithmetic operators (addition, subtraction, division, multiplication, modulus, integer division, exponentiation)	2.5.1	understand the purpose of, and how to use, arithmetic operators (add, subtract, divide, multiply, modulus, integer division)	Low Added	Include exponentiation in your teaching.	This can be taught alongside 1.2.3.
6.5.2	be able to write programs that use relational operators (equal to, less than, greater than, not equal to, less than or equal to, greater than or equal to)	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)	None	Nothing	This is in the context of selection and iteration conditions.
6.5.3	be able to write programs that use logical operators (AND, OR, NOT)	2.5.3	understand the purpose of, and how to use, logic operators (AND, OR, NOT)	None	Nothing	This is in the context of selection and iteration conditions.
6.6.1	be able to write programs that use pre-existing (built-in, library) and user-devised subprograms (procedures, functions)	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	High Added	See the PLS for the list of modules and subprograms that need to be covered.	Procedures and functions were included in the old 2.6.2 anyway so there's no change related to those.
6.6.2	be able to write functions that may or may not take parameters but must return values, and procedures that may or may not take parameters but do not return values	2.6.2	understand the concept of passing data into and out of subprograms (procedures, functions)	Low Clarification	Nothing	Note that Python implements functions and procedures in the same way, but can only be used as a function if a RETURN value is specified.
		2.6.3	be able to create subprograms that use parameters			
6.6.3	understand the difference between and be able to write programs that make appropriate use of global and local variables	2.3.5	understand the need for, and how to use, global and local variables when implementing subprograms	None	Nothing	

Changes in old 1CP1 specification order

Old Topic 1: Problem Solving

1CP1		1CP2		What difference does it make?		
Spec	**Learning Objective**	**Spec**	**Learning Objective**	**Impact Summary**	**What you need to do**	**Qualified opinion**
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)	1.2.1	be able to follow and write algorithms (flowcharts, pseudocode, program code) that use sequence, selection, repetition (count-controlled, condition-controlled) and iteration (over every item in a data structure), and input, processing and output to solve problems	High	Instead of teaching Edexcel's Pseudo-code you need to use their Assessment Reference Language. Loops must include count-controlled (FOR) and pre-conditioned (WHILE) and post-conditioned (UNTIL). Expect loops to iterate through data structures.	There is an error here in that iteration isn't just about data structures but also includes loops. With the PLS being Python, it's easier to teach just the Python language, but there is a wider scope now that needs covering. It will be interesting to see how they test post-conditioned loops in the exam when these can't be used in Python.
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)	1.2.1	be able to follow and write algorithms (flowcharts, pseudocode, program code) that use sequence, selection, repetition (count-controlled, condition-controlled) and iteration (over every item in a data structure), and input, processing and output to solve problems			
1.1.3	Understand the purpose of a given algorithm and how an algorithm works	6.1.2	be able to read, write, analyse and refine programs written in a high-level programming language	Low Clarified	The main change here is the ability to refine existing programs which could be presented electronically as Python files in Paper 2. See also 1CP1 spec 1.1.6 and 2.1.1	This is neatly put all in one place.
1.1.4	Understand how to determine the correct output of an algorithm for a given set of data	1.2.4	be able to determine the correct output of an algorithm for a given set of data and use a trace table to determine what value a variable will hold at a given point in an algorithm	Low Clarified	See 2.1.6 from 1CP1 spec which covers the same thing	
1.1.5	Understand how to identify and correct errors in algorithms	1.2.5	understand types of errors that can occur in programs (syntax, logic, runtime) and be able to identify and correct in algorithms	None	Nothing See also 1CP1 spec 2.1.3	This is just a clarification that students need to be able to identify and correct errors rather than just understand how to do it.
1.1.6	understand how to code an algorithm in a high-level language	6.1.2	be able to read, write, analyse and refine programs written in a high-level programming language	Low Clarified	The main change here is the ability to refine existing programs which could be presented electronically as Python files in Paper 2. See also 1CP1 spec 1.1.3 and 2.1.1	This is neatly put all in one place.
1.1.7	understand how the choice of algorithm is influenced by the data structures and data values that need to be manipulated		Not in Edexcel 1CP2			This was never clear so it's good to see it go.

1CP1		1CP2		What difference does it make?		
Spec	**Learning Objective**	**Spec**	**Learning Objective**	**Impact Summary**	**What you need to do**	**Qualified opinion**
1.1.8	understand how standard algorithms (bubble sort, merge sort, linear search, binary search) work	1.2.6	understand how standard algorithms (bubble sort, merge sort, linear search, binary search) work	None	Nothing	
1.1.9	be able to evaluate the fitness for purpose of algorithms in meeting specified requirements efficiently using logical reasoning and test data	1.2.7	be able to use logical reasoning and test data to evaluate a program's fitness for purpose and efficiency (number of compares, number of passes through a loop, use of memory)	Low Clarified	No need to evaluate an algorithm against its requirements. Cover in teaching how to evaluate efficiency in terms of the number of compares, passes and use of memory. This should be also be done for search and sort methods including best- and worse-case scenarios for searches and being aware that a merge sort is an 'out of place sort' and so requires more memory than a bubble sort	This has been clarified as to what efficiency means. It's confusing that it appears in both 1.2.7 and 6.1.6 if 1CP2 spec. Although it's likely to be taught in unit 6, it's more likely to be tested in Paper 1.
		6.1.6	be able to use logical reasoning and test data to evaluate a program's fitness for purpose and efficiency (number of compares, number of passes through a loop, use of memory)			
1.2.1	be able to analyse a problem, investigate requirements (inputs, outputs, processing, initialisation) and design solutions	1.1.1	understand the benefit of using decomposition and abstraction to model aspects of the real world and analyse, understand and solve problems	Low Removed	No need to cover investigating requirements. Although designing solutions won't be tested, it must still be covered as part of the Programming Project. Decomposition and abstraction are covered in both 1.1.1 and 6.1.1 which means understanding the benefit will be tested in Paper 1 and using decomposition and abstraction will be tested in Paper 2.	Decomposition is about decomposing into smaller problems so that's probably why the text has been removed but it still needs teaching.
1.2.2	be able to decompose a problem into smaller sub-problems	1.1.1	understand the benefit of using decomposition and abstraction to model aspects of the real world and analyse, understand and solve problems			
		6.1.1	be able to use decomposition and abstraction to analyse, understand and solve problems			
1.2.3	understand how abstraction can be used effectively to model aspects of the real world	1.1.1	understand the benefit of using decomposition and abstraction to model aspects of the real world and analyse, understand and solve problems			
		6.1.1	be able to use decomposition and abstraction to analyse, understand and solve problems			
1.2.4	be able to program abstractions of real-world examples	1.1.1	understand the benefit of using decomposition and abstraction to model aspects of the real world and analyse, understand and solve problems			
		6.1.1	be able to use decomposition and abstraction to analyse, understand and solve problems			

Old Topic 2: Programming

1CP1		1CP2		What difference does it make?		
Spec	**Learning Objective**	**Spec**	**Learning Objective**	**Impact Summary**	**What you need to do**	**Qualified opinion**
2.1.1	be able to write programs in a high-level programming language	6.1.2	be able to read, write, analyse and refine programs written in a high-level programming language	Low Clarified	The main change here is the ability to refine existing programs which could be presented electronically as Python files in Paper 2. See also 1CP1 spec 1.1.3 and 1.1.6	This is neatly put all in one place.
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	6.1.4	be able to use techniques (layout, indentation, comments, meaningful identifiers, white space) to make programs easier to read, understand and maintain	Low Clarified	Adapt teaching notes for "descriptive names" to be "meaningful identifiers". Include layout and white space in your teaching.	This is much clearer. Although students don't need to answer questions in an exam about the benefits, it's good for them to be aware of them anyway.
2.1.3	be able to differentiate between types of error in programs (logic, syntax, runtime)	1.2.5	understand types of errors that can occur in programs (syntax, logic, runtime) and be able to identify and correct in algorithms	Low Clarified		
		6.1.5	be able to identify, locate and correct program errors (logic, syntax, runtime)			
2.1.4	be able to design and use test plans and test data (normal, boundary, erroneous)		Not in Edexcel 1CP2			
2.1.5	be able to interpret error messages and identify, locate and fix errors in a program	6.1.5	be able to identify, locate and correct program errors (logic, syntax, runtime)	Low Clarified	Nothing See also 2.1.3 of 1CP1 spec	This is just a clarification that students need to be able to use a trace table rather than just understand how to use one.
2.1.6	be able to determine what value a variable will hold at a given point in a program (trace table)	1.2.4	be able to determine the correct output of an algorithm for a given set of data and use a trace table to determine what value a variable will hold at a given point in an algorithm	Low Clarify	Nothing See also 1.1.4 of 1CP1 spec.	This is just a clarification that students need to be able to use a trace table rather than just understand how to use one.
2.1.7	be able to determine the strengths and weaknesses of a program and suggest improvements	1.2.7	be able to use logical reasoning and test data to evaluate a program's fitness for purpose and efficiency (number of compares, number of passes through a loop, use of memory)		No need to evaluate an algorithm against its requirements. Cover in teaching how to evaluate efficiency in terms of the number of compares, passes and use of memory. See also 1.1.9 of 1CP1 spec	This has been clarified as to what efficiency means. It's confusing that it appears in both 1.2.7 and 6.1.6 if 1CP2 spec. Although it's likely to be taught in unit 6, it's more likely to be tested in Paper 1. The GSG incorrectly references SAM Paper 2 Q4 as being related to 6.1.6 when it is actually related to 6.1.5.
		6.1.6	be able to use logical reasoning and test data to evaluate a program's fitness for purpose and efficiency (number of compares, number of passes through a loop, use of memory)			

1CP1		1CP2		What difference does it make?		
Spec	Learning Objective	Spec	Learning Objective	Impact Summary	What you need to do	Qualified opinion
2.2.1	understand the structural components of a program (variable and ~~type declarations~~, command sequences, selection, iteration, data structures, subprograms)	6.2.1	understand the function of and be able to identify the structural components of programs (constants, variables, initialisation and assignment statements, command sequences, selection, repetition, iteration, data structures, subprograms, parameters, input/output)	Low Added Removed	Include constants, initialisation, and assignment statements in your teaching. There's no need to include type declaration which makes sense as Python is being used. Parameters are covered in 6.6.2.	This is very unclear as to the difference between repetition and iteration. Technically repetition could be writing the same line of code several times, but iteration is generally understood to be repetition of code in a loop so the addition of repetition doesn't help.
2.2.2	be able to use sequencing, selection and iteration constructs in their programs	6.2.2	be able to write programs that make appropriate use of sequencing, selection, repetition (count-controlled, pre-conditioned, post-conditioned), iteration (over every item in a data structure) and single entry/exit points from code blocks and subprograms	Low Confused Added	Although there is an error here, this is tested in a practical sense and so you can teach the term iteration correctly. The inclusion of single entry/exit points from code blocks and subprograms would suggest that Edexcel wouldn't accept use of BREAK in Python, but their GSG states BREAK is acceptable as long as the logic of the code is still clear.	The suggestion that count-controlled, pre-conditioned and post-conditioned loops are repetition and not iteration is wrong. The suggestion that iteration only applies to items in a data structure is wrong.
2.3.1	understand the need for, and understand how to use, data types (integer, real, Boolean, char)	6.3.1	be able to write programs that make appropriate use of primitive data types (integer, real, Boolean, char) and one- and two-dimensional structured data types (string, array, record)	Low Clarification	Ensure teaching of manipulation of strings includes length, position, substrings and case conversion. See the PLS. It would make sense to teach 1.2.2 within topic 6.	The addition of the word primitive doesn't change what needs to be covered. Note that the GSG contains an error that suggests the list data structure when using mixed data types is storing records. This isn't correct as records have fixed lengths and lists do not. It is correct though that records can contain mixed data types.
2.3.2	understand the need for, and understand how to use, data structures (records, one-dimensional arrays, two-dimensional arrays)	1.2.2	understand the need for and be able to follow and write algorithms that use variables and constants and one- and two-dimensional data structures (strings, records, arrays)			
		6.3.1	be able to write programs that make appropriate use of primitive data types (integer, real, Boolean, char) and one- and two-dimensional structured data types (string, array, record)			
2.3.3	understand the need for, and how to manipulate, strings	6.3.1	be able to write programs that make appropriate use of primitive data types (integer, real, Boolean, char) and one- and two-dimensional structured data types (string, array, record)			
		6.3.3	be able to write programs that manipulate strings (length, position, substrings, case conversion)			

1CP1		1CP2		What difference does it make?		
Spec	Learning Objective	Spec	Learning Objective	Impact Summary	What you need to do	Qualified opinion
2.3.4	understand the need for, and how to use, variables and constants	1.2.2	understand the need for and be able to follow and write algorithms that use variables and constants and one- and two-dimensional data structures (strings, records, arrays)	None	In the GSG, Edexcel expect students to use CAPITALS to declare constants, although there is no implementation of constants in Python.	
		6.3.2	be able to write programs that make appropriate use of variables and constants			
2.3.5	understand the need for, and how to use, global and local variables when implementing subprograms	6.6.3	understand the difference between and be able to write programs that make appropriate use of global and local variables	None	Nothing	
2.4.1	understand how to write code that accepts and responds appropriately to user input	6.4.1	be able to write programs that accept and respond appropriately to user input	Low Added	The PLS requires students to be able to use the string.format() method for formatting output.	
2.4.2	understand the need for, and how to implement, validation	6.4.3	understand the need for and be able to write programs that implement validation (length check, presence check, range check, pattern check)	Low Clarification	Ensure all the validation methods listed are covered. See also old spec 5.2.1.	
2.4.3	be able to write code that reads/writes from/to a text file	6.4.2	be able to write programs that read from and write to comma separated value text files	Low Clarification	It was already necessary to teach CSV import and export for the Programming Project, but this means it can now be tested in Paper 2 so ensure students are able to recall how to do this.	
2.5.1	understand the purpose of, and how to use, arithmetic operators (add, subtract, divide, multiply, modulus, integer division)	1.2.3	understand the need for and be able to follow and write algorithms that use arithmetic operators (addition, subtraction, division, multiplication, modulus, integer division, exponentiation), relational operators (equal to, less than, greater than, not equal to, less than or equal to, greater than or equal to) and logical operators (AND, OR, NOT)	Low Added	Include exponentiation in your teaching.	Easiest to teach 1.2.3 alongside topic 6
		6.5.1	be able to write programs that use arithmetic operators (addition, subtraction, division, multiplication, modulus, integer division, exponentiation)			

1CP1		1CP2		What difference does it make?		
Spec	**Learning Objective**	**Spec**	**Learning Objective**	**Impact Summary**	**What you need to do**	**Qualified opinion**
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)	1.2.3	understand the need for and be able to follow and write algorithms that use arithmetic operators (addition, subtraction, division, multiplication, modulus, integer division, exponentiation), relational operators (equal to, less than, greater than, not equal to, less than or equal to, greater than or equal to) and logical operators (AND, OR, NOT)			
		6.5.2	be able to write programs that 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)	1.2.3	understand the need for and be able to follow and write algorithms that use arithmetic operators (addition, subtraction, division, multiplication, modulus, integer division, exponentiation), relational operators (equal to, less than, greater than, not equal to, less than or equal to, greater than or equal to) and logical operators (AND, OR, NOT)			
		6.5.3	be able to write programs that use logical operators (AND, OR, NOT)			
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	1.1.2	understand the benefits of using subprograms	High Added	See the PLS for the list of modules and subprograms that need to be covered.	Procedures and functions were included in the old 2.6.2 anyway so there's no change related to those.
		6.6.1	be able to write functions that may or may not take parameters but must return values, and procedures that may or may not take parameters but do not return values			
2.6.2	understand the concept of passing data into and out of subprograms (procedures, functions)	6.6.2	be able to write functions that may or may not take parameters but must return values, and procedures that may or may not take parameters but do not return values			
2.6.3	be able to create subprograms that use parameters	6.6.2	be able to write subprograms (procedures, functions) using parameters to pass data from a calling program into a subprogram	None	Nothing	

Old Topic 3: Data

1CP1		1CP2		What difference does it make?		
Spec	Learning Objective	Spec	Learning Objective	Impact Summary	What you need to do	Qualified opinion
3.1.1	understand that computers use binary to represent data (numbers, text, sound, graphics) and program instructions	2.1.1	understand that computers use binary to represent data (numbers, text, sound, graphics) and program instructions and be able to determine the maximum number of states that can be represented by a binary pattern of a given length	Medium Added	Teach that 2^n gives the number of unique values that n bits can represent	This is straightforward
3.1.2	understand how computers represent and manipulate numbers (unsigned integers, signed integers (sign and magnitude, two's complement))	2.1.2	understand how computers represent and manipulate unsigned integers and two's complement signed integers	Medium Removed	Remove sign and magnitude method of signed integers from your teaching.	This makes a lot of sense as the two methods were confusing when taught alongside each other at this level.
3.1.3	be able to convert between binary and denary whole numbers (0–255)	2.1.3	be able to convert between denary and 8-bit binary numbers (0 – 255, -127 – 128)	Low Clarified	If not already doing it, ensure students are taught how to convert 8-bit signed binary numbers.	This clarifies that negative binary numbers need to be converted too.
3.1.4	understand how to perform binary arithmetic (add, shifts (logical and arithmetic)) and understand the concept of overflow	2.1.4	be able to add together two positive binary patterns and apply logical and arithmetic binary shifts	Low Added	Cover why a right binary shift results in a lack of precision (DIV rather than ÷)	It's not clear whether students need to understand the additional issues with 2's complement shifts.
		2.1.5	understand the concept of overflow in relation to the number of bits available to store a value			
3.1.5	understand why hexadecimal notation is used and be able to convert between hexadecimal and binary	2.1.6	understand why hexadecimal notation is used and be able to convert between hexadecimal and binary	None		
3.2.1	understand how computers encode characters using ASCII	2.2.1	understand how computers encode characters using 7-bit ASCII	Low Clarified	No need to teach about 8-bit extended ASCII and Unicode in depth but students need to be aware that these provide larger character sets.	
3.2.2	understand how bitmap images are represented in binary (pixels, resolution, colour depth)	2.2.2	understand how bitmap images are represented in binary (pixels, resolution, colour depth)	High Added	The GSG requires an understanding of resolution (ppi) and calculating the physical size of an image. The GSG wants students to be aware that higher resolutions result in higher file sizes. As well as constructing an expression for the file size, students need to be able to calculate a missing value in the equation, e.g. colour depth. They also need to be able to convert binary data into a 2 colour (colour depth of 1 bit) image	Resolution vs image size is a difficult concept so it's a shame it's been included, especially as resolution isn't always directly associated with file size as it's related to output rather than storage.

1CP1		1CP2		What difference does it make?		
Spec	Learning Objective	Spec	Learning Objective	Impact Summary	What you need to do	Qualified opinion
3.2.3	understand how sound, an analogue signal, is represented in binary	2.2.3	understand how analogue sound is represented in binary (amplitude, sample rate, bit depth, sample interval)	Low Added	Ensure you are teaching the terms amplitude, sample rate, bit depth and sample interval. As well as being able to construct an expression for the file size, students need to be able to calculate a missing value in the equation, e.g. bit depth.	Sample interval isn't actually used to represent sound in binary but it's been included in the specification so needs teaching.
3.2.4	understand the limitations of binary representation of data (sampling frequency, resolution) when constrained by the number of available bits	2.2.4	understand the limitations of binary representation of data (sampling frequency, resolution) when constrained by the number of available bits	Low Clarified	It's possible that the bit in brackets is being removed by Edexcel from the specification.	There are some errors in terminology here being inconsistent with 2.2.3 which uses correct terminology. It's also confusing because the bit depth (resolution) is the number of bits per sample which is what the constraint is.

1CP1		1CP2		What difference does it make?		
Spec	Learning Objective	Spec	Learning Objective	Impact Summary	What you need to do	Qualified opinion
3.3.1	understand how to convert between the terms 'bit, nibble, byte, kilobyte (KB), megabyte (MB), gigabyte (GB), terabyte (TB)'	2.3.1	understand that data storage is measured in binary multiples (bit, nibble, byte, kibibyte, mebibyte, gibibyte, tebibyte) and be able to construct expressions to calculate file sizes and data capacity requirements	Low Clarified Added	Ensure you teach the difference between kilobytes (1000) and kibibytes (1024). Cover the use of expressions to show how a file size would be calculated for text files, image files and sound files. 1024 must be used, not 1000.	The use of kibibytes etc will be because Edexcel were wrongly using 1024 in the exam for kilobytes which are actually 1000. However, if they had stuck to kilobytes etc then calculations could be made instead of expressions. Although expressions weren't on the 2012 specification, they were expected in the exams.
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	2.3.2	understand the need for data compression and methods of compressing data (lossless, lossy)	Medium Removed	No need to teach examples of JPEG, MPE and RLE but examples are usually helpful. The GSG says pros and cons of lossy vs lossless should be covered.	This means specific questions in the exam won't be asked about these examples, but higher ability students can still be stretched by using examples.
3.3.3	understand how a lossless, run-length encoding (RLE) algorithm works	2.3.2	understand the need for data compression and methods of compressing data (lossless, lossy)			
3.3.4	understand that file storage is measured in bytes and be able to calculate file sizes	2.3.1	understand that data storage is measured in binary multiples (bit, nibble, byte, kibibyte, mebibyte, gibibyte, tebibyte) and be able to construct expressions to calculate file sizes and data capacity requirements	Low Clarified Added	See 3.3.1	
3.4.1	understand the need for data encryption	5.3.2	understand methods of protecting digital systems and data (anti-malware, encryption, acceptable use policies, backup and recovery procedures)	Low Clarified Removed	See 5.2.5 of 1CP1 spec	
3.4.2	understand how a Caesar cipher algorithm works					
3.5.1	understand the characteristics of structured and unstructured data		Not in Edexcel 1CP2			
3.5.2	understand that data can be decomposed, organised and managed in a structured database (tables, records, fields, relationships, keys)					

Old Topic 4: Computers

1CP1		1CP2		What difference does it make?		
Spec	**Learning Objective**	**Spec**	**Learning Objective**	**Impact Summary**	**What you need to do**	**Qualified opinion**
4.1.1	understand the input-process-output model	1.2.1	be able to follow and write algorithms (flowcharts, pseudocode, program code) that use sequence, selection, repetition (count-controlled, condition-controlled) and iteration (over every item in a data structure), and input, processing and output to solve problems	Low Removed	Although there's no need to cover input-process-output model, it does come in handy. For the rest see old specification 1.1.2.	
4.2.1	understand the function of the hardware components of a computer system (CPU, main memory, secondary storage, input and output devices) and how they work together	3.1.1	understand the von Neumann stored program concept and the role of main memory (RAM), CPU (control unit, arithmetic logic unit, registers), clock, address bus, data bus, control bus in the fetch-decode-execute cycle	Low Removed	No need to teach input and output devices No need to understand how devices work together No need to cover ROM and cache Need to know the purpose of registers but no need to name them or describe their function.	Input and output devices are very basic, but they were good for Grade 1 and 2 students.
		3.1.2	understand the role of secondary storage and the ways in which data is stored on devices (magnetic, optical, solid state)	Low Clarified	Ensure you teach about how data is stored physically on magnetic, optical and solid state devices Include in your teaching the role/purpose of secondary storage (i.e. non-volatile storage). No need to cover 'cloud' storage. Students need to justify why one type of storage may be more suitable for a given purpose.	Data stored in the cloud is still stored on physical devices so it makes sense to remove it. The role of secondary storage is important as it's non-volatile unlike RAM. "How" instead of "ways in which" clarifies that students need to understand how data is stored on each medium.
4.2.2	understand the function of different types of main memory (RAM, ROM, cache)	3.1.1	understand the von Neumann stored program concept and the role of main memory (RAM), CPU (control unit, arithmetic logic unit, registers), clock, address bus, data bus, control bus in the fetch-decode-execute cycle	Low Removed	No need to cover ROM and cache	ROM and cache were always good differentiators for higher ability students.
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-decode-execute cycle (the Von Neumann model)	3.1.1	understand the von Neumann stored program concept and the role of main memory (RAM), CPU (control unit, arithmetic logic unit, registers), clock, address bus, data bus, control bus in the fetch-decode-execute cycle	None	Nothing See 4.2.2 above also	

1CP1		1CP2		What difference does it make?		
Spec	Learning Objective	Spec	Learning Objective	Impact Summary	What you need to do	Qualified opinion
4.2.4	understand how data is stored on physical devices (magnetic, optical, solid state)	3.1.2	understand the role of secondary storage and the ways in which data is stored on devices (magnetic, optical, solid state)	Low Clarified	Ensure you teach about how data is stored on magnetic, optical and solid state devices Include in your teaching the role/purpose of secondary storage (i.e. non-volatile storage) No need to cover 'cloud' storage	Data stored in the cloud is still stored on physical devices so it makes sense to remove it. The role of secondary storage is important as it's non-volatile unlike RAM. "How" instead of "ways in which" clarifies that students need to understand how data is stored on each medium.
4.2.5	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	3.1.3	understand the concept of an embedded system and what embedded systems are used for	Low Clarified Added	GSG states students should understand the role of a microcontroller and be familiar with the IoT and be aware of privacy and security concerns associated with it.	
4.3.1	be able to construct truth tables for a given logic statement (AND, OR, NOT)	1.3.1	be able to apply logical operators (AND, OR, NOT) in appropriate truth tables with up to three inputs to solve problems	Low Removed	No need to teach logic statements. Ensure you are giving students plenty of practice at using truth tables for single and multiple logical operators	Although they don't need to be taught, it's useful to understand how logic statements work in a truth table in order to be able to apply them to different situations.
4.3.2	be able to produce logic statements for a given problem		Not in Edexcel 1CP2			
4.4.1	know what an operating system is and how it manages files, processes, hardware and the user interface	3.2.1	understand the purpose and functionality of an operating system (file management, process management, peripheral management, user management)	Medium Added Removed	No need to cover the user interface as a function of an operating system. Ensure your teaching now also includes peripheral management and user management as functions of the operating system. Ensure your teaching includes the purpose of an operating system.	

1CP1		1CP2		What difference does it make?		
Spec	Learning Objective	Spec	Learning Objective	Impact Summary	What you need to do	Qualified opinion
4.4.2	understand the purpose and functions of utility software (managing, repairing and converting files; compression; defragmentation; backing up; anti-virus, anti-spyware)	3.2.2	understand the purpose and functionality of utility software (file repair, backup, data compression, disc defragmentation, anti-malware)	Medium Clarified Removed	No need to cover managing and converting of files. Now need to cover more methods of anti-malware which will include ransomware and keyloggers as suggested in 5.3.1 of 1CP2 spec.	A bit of cross-referencing finds the list of anti-malware in 5.3.1 of 1CP2 spec but it would have been useful here in the specification.
4.4.3	understand how software can be used to simulate and model aspects of the real world		Not in Edexcel 1CP2			
4.5.1	understand what is meant by high-level and low-level programming languages and understand their suitability for a particular task	3.3.1	understand the characteristics and purposes of low-level and high-level programming languages	Low Clarified	Nothing as this would probably be taught anyway.	Although the part about understanding suitability for a particular task has been removed, it's difficult to teach the difference without giving examples of how they would be used.
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.	3.3.2	understand how an interpreter differs from a compiler in the way it translates high-level code into machine code	Low Removed	No need to teach about assemblers. Any difference, not just advantages and disadvantages, should be covered for compilers and interpreters.	This makes it much clearer.

Old Topic 5: Communication and the internet

1CP1		1CP2		What difference does it make?		
Spec	Learning Objective	Spec	Learning Objective	Impact Summary	What you need to do	Qualified opinion
5.1.1	understand why computers are connected in a network	4.1.1	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)	4.1.2	understand different types of networks (LAN, WAN)			
5.1.3	understand wired and wireless connectivity	4.1.4	understand how the characteristics of wired and wireless connectivity impact on performance (speed, range, latency, bandwidth)	Medium Added	Cover how the characteristics of both wired and wireless networks impact on performance. Wired should include fibre and copper.	This is a sensible addition.
5.1.4	understand that network data speeds are measured in bits per second (Mbps, Gbps)	4.1.5	understand that network speeds are measured in bits per second (kilobit, megabit, gigabit) and be able to construct expressions involving file size, transmission rate and time	Low Clarified Added	Teach students how to create expressions for calculating transmission rates. Ensure this section is taught using base-10 units and is based on bits (not bytes) and 2.3.1 is taught using base 2 and is based on bytes.	Edexcel had already been asking students to create expressions in exams even though it wasn't in the specification. The use of kilobit, megabit and gigabit is clear, but it may be confusing now kilobyte etc has been replaced with kibibyte etc in 2.3.1 meaning that students may get confused between base 2 in 2.3.1 and base 10 in 4.1.5.
5.1.5	understand the role of and need for network protocols (Ethernet, Wi-Fi, TCP/IP, HTTP. HTTPS, FTP, email (POP3, SMTP, IMAP))	4.1.6	understand the role of and need for network protocols (Ethernet, Wi-Fi, TCP/IP, HTTP, HTTPS, FTP and email protocols (POP3, SMTP, IMAP))	None	No need to know individual protocols that make up the Ethernet and Wi-Fi 'families'.	The removal of the word network makes no difference.
5.1.6	understand that data can be transmitted in packets using layered protocol stacks (TCP/IP)	4.1.7	understand how the 4-layer (application, transport, internet, link) TCP/IP model handles data transmission over a network	Medium Clarified	Continue to teach the TCP/IP model including packet switching which is used at the transport layer. Ensure other layers are also covered and how data moves up and down the stack.	
5.1.7	understand characteristics of network topologies (bus, ring, star, mesh)	4.1.8	understand characteristics of network topologies (bus, star, mesh)	Low Removed	No need to cover ring networks.	This is a welcome change as ring networks are very rare. Perhaps removing bus networks would also have been a good idea.

1CP1		1CP2		What difference does it make?		
Spec	Learning Objective	Spec	Learning Objective	Impact Summary	What you need to do	Qualified opinion
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)	4.2.1	understand the importance of network security, ways of identifying network vulnerabilities (penetration testing, ethical hacking) and methods of protecting networks (access control, physical security, firewalls)	Low Clarified	See 1CP1 spec 5.2.4 for comparison of 1CP2 spec 4.2.1	
		6.4.3	understand the need for and be able to write programs that implement validation (length check, presence check, range check, pattern check)	Low Clarified	Validation and authentication are still covered but in 1CP2 spec 6.4.3 and 6.4.4 so will only be tested in Paper 2. Ensure all the validation methods listed are covered.	It's clear which validation techniques students need to cover.
		6.4.4	understand the need for and be able to write programs that implement authentication (ID and password, lookup)	Low Clarified	Ensure that you are including ID, password and lookup of ID/password in a list in your teaching for implementation methods using Python. Differentiate as lower ability students will find lookup difficult.	This clarifies this much better. The old examples in brackets of authentication techniques were confusing so calling them methods of protecting networks is much better.
5.2.2	understand security issues associated with the 'cloud' and other contemporary storage	5.3.1	understand the threat to digital systems posed by malware (viruses, worms, Trojans, ransomware, key loggers) and how hackers exploit technical vulnerabilities (unpatched software, out-of-date anti-malware) and use social engineering to carry out cyberattacks	High Clarified	Cover the digital threats as listed. Prevention methods are now all in 5.2.2. Social engineering now covers pretexting, phishing, baiting and quid pro quo techniques. No longer need to cover threats from USB devices and eavesdropping.	This is much neater to have cyber threats all in one place.
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					
5.2.4	understand methods of identifying vulnerabilities including penetration testing, ethical hacking, commercial analysis tools and review of network and user policies	4.2.1	understand the importance of network security, ways of identifying network vulnerabilities (penetration testing, ethical hacking) and methods of protecting networks (access control, physical security, firewalls)	Low Removed	No need to cover commercial analysis tools and review of network and user policies. See 1CP1 spec 5.2.1	This clarifies this much better. The old examples in brackets of authentication techniques were confusing so calling them methods of protecting networks is much better.

1CP1		1CP2		What difference does it make?		
Spec	**Learning Objective**	**Spec**	**Learning Objective**	**Impact Summary**	**What you need to do**	**Qualified opinion**
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	3.2.3	understand the importance of developing robust software and methods of identifying vulnerabilities (audit trails, code reviews)	Medium Removed	No need to cover "securing operating systems" or modular testing. See GSG for additional guidance related to robust software. Cover the protection methods as listed. No longer need to cover the Caesar cipher algorithm, but it's still worth teaching as it helps to explain encryption well. Backup and recovery procedures should include RAID, off-site storage and standby equipment/premises. It's not necessary to cover specific backup methods. All security protection methods are now neatly in the new 1CP2 spec 5.3.2 with exception of network security in 4.2.1 and using validation and authentication in 6.4.3 and 6.4.4	This is much neater to have security measures all in one place.
		5.3.2	understand methods of protecting digital systems and data (anti-malware, encryption, acceptable use policies, backup and recovery procedures)			
5.3.1	understand what is meant by the internet and how the internet is structured (IP addressing, routers)	4.1.3	understand how the internet is structured (IP addressing, routers)	Low Removed	No need to know 'what is meant by the internet'. GSG states students need to know about packet switching and packet headers.	Although this has been removed, it would be impossible to teach this without an understanding of what the internet is.
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)		Not in Edexcel 1CP2			

Old Topic 6: The bigger picture

1CP1		1CP2		What difference does it make?		
Spec	**Learning Objective**	**Spec**	**Learning Objective**	**Impact Summary**	**What you need to do**	**Qualified opinion**
6.1.1	understand the environmental impact of technology (health, energy use, resources) on society	5.1.1	understand environmental issues associated with the use of digital devices (energy consumption, manufacture, replacement cycle, disposal)	High Removed Added	No longer need to cover health when teaching about environmental issues. Need to cover manufacture, replacement cycle and disposal for environmental issues including how to reduce the environmental impact.	The additions make sense. Although health has been removed, the disposal and replacement cycle (and thus recycling) does have an effect on health, particularly in 3rd world countries.
6.1.2	understand the ethical impact of using technology (privacy, inclusion, professionalism) on society	5.2.1	understand ethical and legal issues associated with the collection and use of personal data (privacy, ownership, consent, misuse, data protection)	High Changed	No need to cover inclusion and professionalism. Focus is now on personal data and should include ownership, consent, misuse and data protection. The UK Data Protection Act 2018 should be covered including the principals, right to be forgotten, rights of data subjects and obligations of data users. Students should be aware of the Privacy and Electronic Communications Regulations (PECR) in relation to cookies for collecting personal data. Students should be aware of how the Computer Misuse Act helps to protect individuals' personal data by deterring hackers.	This is good because it narrows down the focus to issues to do with personal data rather than any sort of technology.
		5.2.2	understand ethical and legal issues associated with the use of artificial intelligence, machine learning and robotics (accountability, safety, algorithmic bias, legal liability)	High Added	This is a new section that needs teaching, but it is focused rather than open ended. Robotics could include autonomous machines, surgical robots, driverless cars, automated weapons and assistive robotics.	This is good because it narrows down the focus to issues to do with AI, machine learning and robotics rather than any sort of technology.
6.1.3	understand the legal impact of using technology (intellectual property, patents, licensing, open source and proprietary software, cyber-security) on society	5.2.3	understand methods of intellectual property protection for computer systems and software (copyright, patents, trademarks, licencing)	High Added	Cyber security is covered in 5.3.1 and 5.3.2 so no need to relate it to this section. The focus is now on intellectual property rather than legal impacts on a range of aspects. Open source and proprietary software should still be covered as methods of licensing. In addition, copyright, patents and trademarks need to be covered including the difference between them. There's no need to know about specific legislation but there should be an awareness it exists.	This is much more succinct although it would be useful if licensing was more clearly defined.