STMLC	Year Group 10-11 GCSE
Subject: Computer Science	Exam board and specification AQA

Specification reference	Specification content	Learning outcomes	Suggested timing (lessons)	Teaching and learning activities
Autumn term 3.4 Computer systems Total teaching time: 6 hours 3.4.2	Construct truth tables for NOT, AND, OR gates, Construct truth tables for simple logic circuits and interpret them. Create simple logic circuit diagrams.	Be able to construct truth tables for gates and circuits. Be able to draw logic circuits to represent a simple logic problem.	2 hours	Consider the basic operations of AND, OR and NOT (students may have already come across these in the context of programming or databases depending on the order in which the sections are taught). Look at truth tables for each gate. Draw a logic circuit and then build a truth table for it. Students should then try some exercises completing truth tables for different logic circuits. Introduce the idea of drawing a logic circuit to represent a specific problem. Students should then try to draw logic circuits for a few

Specification reference	Specification content	Learning outcomes	Suggested timing (lessons)	Teaching and learning activities
				problems. This could be done on paper or using an online logic circuit simulator or physically using electronics or tools such as logic goats. It is not required for the specification but it would be useful to link this in to hardware and the design of the processor by explaining how gates can be combined to make a processor or memory.
3.4.4	Explain Von Neumann architecture. Explain role of main memory, components of CPU, buses. Understand and explain the fetch-execute cycle.	Explain Von Neumann architecture. Explain role of main memory, components of CPU, buses. Understand and explain the fetch-execute cycle.	1 hour	A good way to introduce this is to have old PCs that students can look inside of to identify the component parts. This could be done with photographs but having real PCs makes it more interesting. The role of the components needs to be explained. Students only need a high-level understanding of the fetch-execute cycle. They don’t need to know the details of register operations etc. A range of online simulators can be used to illustrate this.

Specification reference	Specification content	Learning outcomes	Suggested timing (lessons)	Teaching and learning activities
3.4.4	Understand difference between main memory and secondary storage and between RAM and ROM. Be able to explain volatile and non-volatile. Explain the effect of clock speed, number of cores and cache size on processor performance.	Understand difference between main memory and secondary storage and between RAM and ROM. Be able to explain volatile and non-volatile. Explain the effect of clock speed, number of cores and cache size on processor performance.	0.5 hours	This is not a very practical topic. Most of the content is probably best explained to the students by the teacher, although students could be asked to research parts of it eg what cache is and how it improves performance. With regard to RAM and ROM, it is helpful to focus on their uses.
3.4.4	Be aware of why secondary storage is needed and the	Explain the operation of solid state, optical and magnetic storage.	1.5 hours	It is useful to have physical devices for students to look at here – a disassembled hard disk drive and CD-ROM drive or similar. There is less of interest that can be seen inside a solid state drive.

Specification reference	Specification content	Learning outcomes	Suggested timing (lessons)	Teaching and learning activities
	different types of secondary storage. Explain the operation of solid state, optical and magnetic storage. Discuss their relative advantages. Explain what cloud storage is and compare it to local storage.	Discuss their relative advantages. Explain what cloud storage is and compare it to local storage.		There are also lots of animations available on the Internet on websites such as howstuffworks.com, which illustrate the principles behind the operation of these devices. Students could make a presentation to explain how each device works. The relative advantages of the devices should be considered in relation to criteria such as maximum capacity, cost per megabyte, robustness, power consumption and portability. Many students will be familiar with using cloud storage such as OneDrive or Apple or Google's cloud storage systems so this aspect of the specification would work well as a discussion with students explaining what they use it for and considering the practical benefits they have seen themselves, but also the risks.
3.4.4	Understand the term 'embedded system' and explain how an embedded system	Explain what an embedded system is and how an embedded system differs from a non-embedded system.	0.5 hours	This is a relatively small topic. Students need to understand that many computer systems are embedded in other devices and the constraints and differences that this produces when compared with non-embedded systems.

Specification reference	Specification content	Learning outcomes	Suggested timing (lessons)	Teaching and learning activities
	differs from a non-embedded system.	Give examples of embedded systems.		Students could be given some scenarios (eg washing machine) and be asked to consider what functionality the system would need and why a non-embedded system would not be suitable. Differences such as processor speed, amount and type of main memory, secondary storage, input and output devices and upgradeability could be considered.
3.4.1, 3.4.3	Define the terms hardware and software and understand the relationship between them. Explain what is meant by systems software and application software and be able to give examples of them.	Define the terms hardware and software and understand the relationship between them. Explain what is meant by systems software and application software and be able to give examples of them. Understand the need for and functions of the OS and utility programs.	0.5 hours	This is very much a theory topic so is probably best delivered by the teacher talking and discussing with the class. For the first point, students simply need to know that hardware is that the electronic or electro-mechanical components of the computer and that software are the programs that run on the hardware and tell it what to do to perform a task. Students need to know that application software completes user-oriented tasks that the user would need to do with or without a computer whereas system software performs tasks related to the management of the computer system.

Specification reference	Specification content	Learning outcomes	Suggested timing (lessons)	Teaching and learning activities
	Understand the need for and functions of the OS and utility programs.			Students need to know that the OS manages processor(s), memory, I/O devices, applications and security but do not need to know how. A utility is a program that helps manage a computer but is not core to its operation eg a compression program, a virus-checker. It might be useful to make students aware that utilities are increasingly being bundled with the OS.

3.1 Algorithms and 3.2 Programming

Total teaching time: 50 hours

Specification reference	Specification content	Learning outcomes	Suggested timing (lessons)	Possible teaching and learning activities	Resource	Exam 'hints and tips'
3.1.1, 3.2.1, 3.2.2, 3.2.3, 3.2.7	Understand and use string, integer and real data types appropriately. Understand how variable declaration and assignment can be used in programs. Be able to use addition, subtraction, multiplication and real division. Be able to perform input and output.	Apply the listed programming techniques. Choose appropriate data types. Use meaningful identifier names and know why. Understand what an algorithm is and the difference between an algorithm and program.	3 hours	Students should be introduced to basic input and output commands, declaring variables (if required by language), and using arithmetic operations. Students will also need to be taught basic aspects of the IDE for their programming language eg how to run a program, how to load/save, how error messages are presented and what they mean. Students should be introduced to the idea of an algorithm and that a program is an implementation of an algorithm.	Notes and videos introducing algorithms and example uses Notes on variables and data types (and some other concepts not required until later).	

Specification reference	Specification content	Learning outcomes	Suggested timing (lessons)	Possible teaching and learning activities	Resource	Exam 'hints and tips'
	Use meaningful identifier names and know why it is important to use them. Understand and explain the term algorithm.			Exercises could include: • getting the computer to display "Hello World" • getting the user to type in their name and outputting hello to them (possibly concatenating forename and surname input separately) • doing simple calculations, for example adding three numbers, multiplying two numbers together • doing more complex calculations, for example area of a rectangle, area of a		

Specification reference	Specification content	Learning outcomes	Suggested timing (lessons)	Possible teaching and learning activities	Resource	Exam ‘hints and tips’
				triangle, area of a circle, area of a trapezium.		
3.2.2, 3.2.4, 3.2.5, 3.2.12	Be able to use selection (if, else, else if, case/switch if appropriate) Be able to use a range of relational operators. Be familiar with and able to use NOT, AND, OR. Using nested selection structures.	Apply the programming techniques listed. Choose appropriate test data to use to check their programs.	3 hours	The focus in this section is on the use of selection statements to determine the path of code execution. Exercises should build in difficulty, starting with simple Yes/No answers using just an If statement then building in complexity in terms of the number of possible outcomes and the complexity of the criteria used. Psuedo-code and flowcharts could be used to illustrate some algorithms which students	Notes and video on use of selection statements Notes on use of AND, OR and NOT	

Specification reference	Specification content	Learning outcomes	Suggested timing (lessons)	Possible teaching and learning activities	Resource	Exam 'hints and tips'
	Be able to select suitable test data that covers normal (typical), boundary and erroneous data. Be able to justify the choice of test data. Be able to understand pseudo-code and flowcharts.			could then write program code for. Whilst completing these exercises, consideration should be given to choosing test data, which is particularly important in boundary situations of which there are many in these exercises. Exercises could include: • exam mark pass/fail • determining if a person is a child/adult/pensioner based on their age • allocating an exam grade based on mark ranges		

Specification reference	Specification content	Learning outcomes	Suggested timing (lessons)	Possible teaching and learning activities	Resource	Exam 'hints and tips'
				• identifying the biggest of two or three numbers • identifying if a triangle is scalene, isosceles or equilateral • classifying the temperature based on a range eg 0 or below = freezing, above 0 but 10 or below = warm.		
3.2.2	Be able to use definite iteration. Be able to use nested iteration.	Be able to use definite iteration.	4 hours	Students should be introduced to the concept of definite iteration and a loop counter. Pseudo-code and flowcharts could be used to illustrate algorithms.	Notes and videos on use of iteration	

Specification reference	Specification content	Learning outcomes	Suggested timing (lessons)	Possible teaching and learning activities	Resource	Exam 'hints and tips'
				Exercises could include: • counting from one to ten • displaying a times table, or all times tables • adding up five numbers (average the same numbers and identify the highest and lowest) • working out factors of a number using brute-force approach • identifying prime numbers using brute-force approach.		

Specification reference	Specification content	Learning outcomes	Suggested timing (lessons)	Possible teaching and learning activities	Resource	Exam ‘hints and tips’
3.1.1, 3.2.2, 3.2.8, 3.2.9, 3.2.12	Be able to use indefinite iteration with conditions at start and end of loop. Be able to use random number generation. Be able to use some string handling techniques. Be able to write simple data validation routines. Be able to write simple	Apply the programming techniques listed. Be able to write simple authentication and validation routines. Understand what abstraction is.	4 hours	Students need to be taught about indefinite iteration and how to use this in their programming language. For students using Python which does not have a post-conditioned loop, they should be taught how to implement post-conditioned loops as equivalent pre-conditioned loops. Students also need to know how to express these types of loop as pseudo-code and flowcharts. As students are now starting to tackle more complex problems, the concept of abstraction, ie	Video on abstraction	

Specification reference	Specification content	Learning outcomes	Suggested timing (lessons)	Possible teaching and learning activities	Resource	Exam ‘hints and tips’
	authentication routines. Understand and explain the term abstraction.			removing unnecessary details from a problem, could be introduced at this point. Exercises could include: • performing simple validation eg that a typed value falls within a range or that an entered value cannot be left blank or is shorter than a minimum length • adding up a sequence of numbers of unknown length • asking users to enter a password until the correct password is		

Specification reference	Specification content	Learning outcomes	Suggested timing (lessons)	Possible teaching and learning activities	Resource	Exam 'hints and tips'
				entered, displaying suitable messages • guessing randomly chosen number until they guess correctly, with clues given about whether guess is too high/low • rolling two dice until a double six is scored, counting how many goes this takes. • throwing darts and getting a random score on board (game starts at a total and plays with the total decreased by each dart thrown until 0 is achieved).		

Specification reference	Specification content	Learning outcomes	Suggested timing (lessons)	Possible teaching and learning activities	Resource	Exam ‘hints and tips’
3.1.1, 3.1.2, 3.1.3, 3.1.4, 3.2.6	Understand the concept of data structures. Use one-dimensional arrays (or equivalent) in the design of solutions to simple problems. Understand that more than one algorithm can be used to solve the same problem. Compare the efficiency of algorithms.	Be able to use one-dimensional arrays. Understand the searching and sorting algorithms listed and be able to compare their efficiency.	5 hours	Students need to be introduced to the concept of a one-dimensional array and should have the opportunity to solve problems using them. They should also cover the four searching and sorting algorithms and have the opportunity to code them, except the merge sort. Before coding these algorithms, it would be helpful for students to look at them in pseudo-code and to trace their execution in a trace table to ensure that they understand how they function. Exercises could include:	Notes on data structures and arrays Video on searching algorithms Video on sorting algorithms Video of bubble sort using Lego bricks Video on bubble sort Simple explanation of Merge Sort video, suitable for GCSE level	Questions will use arrays with a first index of 0 (not 1).

Specification reference	Specification content	Learning outcomes	Suggested timing (lessons)	Possible teaching and learning activities	Resource	Exam 'hints and tips'
	Understand and explain how linear and binary search algorithms work and compare them. Understand and explain how bubble and merge sort algorithms work and compare them. Use trace tables.			• inputting a list of names (or other data) and redisplaying them • inputting a list of parcel weights (total the weights and work out the average, lowest and highest weight) • searching a dictionary to check whether a word is in it using the linear search method • improving the dictionary program to use the binary search method • using the bubble sort algorithm to sort data (eg names) in an array	Video comparing linear and binary search	

Specification reference	Specification content	Learning outcomes	Suggested timing (lessons)	Possible teaching and learning activities	Resource	Exam ‘hints and tips’
				• looking theoretically at how the merge sort algorithm would perform the same sort (implementing merge sort is beyond GCSE but more able students could attempt this) • comparing the efficiency of the search and sort algorithms • representing a game of snakes and ladders using a one-dimensional array to indicate the positions of snakes and ladders.		

Specification reference	Specification content	Learning outcomes	Suggested timing (lessons)	Possible teaching and learning activities	Resource	Exam 'hints and tips'
3.1.1, 3.2.2, 3.2.3, 3.2.10, 3.2.11	Understand and explain the term decomposition. Describe the structured approach to programming. Explain the advantages of the structured approach. Understand the concept of subroutines and be able to use them in programs, including the use of local variables.	Be able to decompose problems into subroutines and call them and know why this is a good idea. Be able to perform integer division, including the use of remainders.	3 hours	Students should be taught about why, when writing longer programs, it is useful to decompose them, and the facilities in their programming language to do this. They should also cover the difference between local and global variables. At this stage, parameters and return values can be ignored. Exercises could include: • making a maths toolkit, with a menu that is used to call different subroutines to work out (for example) the area of different shapes	Video explaining the concept of decomposition Notes and video on decomposition, including the use of parameters (which is not required until later)	

Specification reference	Specification content	Learning outcomes	Suggested timing (lessons)	Possible teaching and learning activities	Resource	Exam ‘hints and tips’
	Explain the advantages of using subroutines in programs. Integer division, including remainders.			• making a program that will allow conversion of numbers between different number bases, with different functions being used for different conversions eg binary to decimal In all subsequent programs, students should be encouraged to consider how the programs can be decomposed into functions.		
3.2.1, 3.2.8, 3.2.11	Use a structured approach to programming, in particular focussing on the use of	Be able to use well-defined interfaces to subroutines, using	6 hours	Emphasis should be on passing input to the functions as parameters and using return to pass values back to the calling program. Input/output via the	Notes and video on decomposition	

Specification reference	Specification content	Learning outcomes	Suggested timing (lessons)	Possible teaching and learning activities	Resource	Exam 'hints and tips'
	parameters and return values. Use a range of string handling operations. Use the char and Boolean data types.	parameters and return values. Use string handing operations and the char data type.		keyboard/screen should not happen within the functions. Students should be taught why this is important, for example in terms of being able to develop and test modules independently and reuse code. Exercises could include: • developing a function that returns the highest of two numbers and adapting this to find the highest of three numbers or to perform other mathematical operations		

Specification reference	Specification content	Learning outcomes	Suggested timing (lessons)	Possible teaching and learning activities	Resource	Exam 'hints and tips'
				• developing a function that indicates whether a number is even or not • developing a function that works out n factorial (n!) • developing a function that returns a string that has been encrypted using the Caesar Cipher with a key selected by the user and adding a decryption function • developing a function to convert a string into Morse code • developing a function that will return a true/false value,		

Specification reference	Specification content	Learning outcomes	Suggested timing (lessons)	Possible teaching and learning activities	Resource	Exam 'hints and tips'
				indicating if two words are anagrams of each other • developing a function that, when sent a number, will return a true/false value indicating whether the number is a perfect number or not and using this in a program to search for perfect numbers using brute-force. In all subsequent programs, students should be encouraged to consider how the programs can be decomposed into functions with interfaces that		

Specification reference	Specification content	Learning outcomes	Suggested timing (lessons)	Possible teaching and learning activities	Resource	Exam 'hints and tips'
				use parameters and return values.		
3.2.7	Be able to read/write from/to a text file.	Be able to use text files for permanent storage of data.	4 hours	Students first need to understand what a text file is. They should then modify some of the previous programs that they have written to read input from/ save output to a text file. Some suitable programs to modify would be: Snakes and ladder positions could be stored in a text file (allowing for the possibility of different boards).		

Specification reference	Specification content	Learning outcomes	Suggested timing (lessons)	Possible teaching and learning activities	Resource	Exam 'hints and tips'
				The dictionary that is searched could be stored in a text file. The sorting program could load the list to sort from a text file and save the sorted list to a different text file. Students should consider how to deal with possible issues such as saving over an existing file or being asked to load a file that does not exist. This would be a point where exception handling could be considered.		
3.2.2, 3.2.6	Use two-dimensional arrays (or equivalent) in the	Use two-dimensional arrays, nested iteration and constants.	10 hours	Students should have the opportunity to write programs using two-dimensional arrays. They will need to	Brief notes on two-dimensional arrays	

Specification reference	Specification content	Learning outcomes	Suggested timing (lessons)	Possible teaching and learning activities	Resource	Exam 'hints and tips'
	design of solutions to simple problems. Use nested iteration. Use of constants.			consider/design how the arrays can be used to represent the problem. Data stored in a two-dimensional array is usually displayed most conveniently using nested loops. A range of games can be readily implemented using two-dimensional arrays. If students have not yet encountered constants, they could be introduced here, for example, to store the size of a game board. Exercises could include: • snakes and ladders • noughts and crosses		

Specification reference	Specification content	Learning outcomes	Suggested timing (lessons)	Possible teaching and learning activities	Resource	Exam ‘hints and tips’
				• battleships.		
3.2.6, 3.2.12	Use records (or equivalent) in the design of solutions to simple problems. Be able to write simple authentication routines.	Use records. Be able to write simple authentication routines.	4 hours	Students should be introduced to the concept of records and why logically grouping related data together is a sensible approach. Exercises could include: • adapt the dictionary program that was written earlier to store equivalent words in two languages in an array of records and perform translation between them		Students using Python can use different methods for representing records. The approach that will be used in the programming exam is detailed in section 3.2.6 of the specification.

Specification reference	Specification content	Learning outcomes	Suggested timing (lessons)	Possible teaching and learning activities	Resource	Exam ‘hints and tips’
				• write an address book program, or a program to keep track of any other data (this data could be saved/loaded from a text file using CSV format) • write a login system with usernames and passwords stored in a file and then loaded into an array of records.		
3.1.1	Use a systematic approach to problem solving and algorithm creation representing those algorithms using	Students can understand algorithms expressed in pseudo-code and flowcharts and use these methods to write algorithms.	2 hours	Throughout learning to program, students should be exposed to how algorithms can be expressed using pseudo-code or flowcharts.		

Specification reference	Specification content	Learning outcomes	Suggested timing (lessons)	Possible teaching and learning activities	Resource	Exam 'hints and tips'
	pseudo-code and flowcharts. Explain simple algorithms in terms of their inputs, processing and outputs. Determine the purpose of simple algorithms.	They can trace the execution of algorithms using a trace table and identify the purpose of an algorithm. They can identify the inputs and outputs of an algorithm together with the required processing,		Students need to have some practice at being able to understand and write algorithms using these methods. They also need to be able to use trace tables to record the values of variables as an algorithm is stepped through and to be able to identify the purpose of an algorithm by tracing it. These skills will be assessed in the exam. It is useful to teach them in parallel with learning to program (perhaps as homework exercises) but it could also be worth giving students the		

Specification reference	Specification content	Learning outcomes	Suggested timing (lessons)	Possible teaching and learning activities	Resource	Exam 'hints and tips'
				opportunity to consolidate their ability to apply these skills. Students should complete exercises where they have to read and write pseudo-code and flowcharts, complete trace tables and deduce the purpose of algorithms.		
3.2.13	Know that there are different levels of programming language: low-level, high-level and explain the main differences between them.	Students should understand the differences between low and high-level languages. They should know the differences between machine code and assembly language.	2 hours	Students only need a theoretical understanding of this topic, so this topic would be best delivered by the teacher as a presentation or through notes or a video with students given the opportunity to answer questions on it. Including real	Online notes covering some of these topics An assembly language tutorial video for a raspberry Pi	

Specification reference	Specification content	Learning outcomes	Suggested timing (lessons)	Possible teaching and learning activities	Resource	Exam 'hints and tips'
	Know that machine code and assembly language are considered to be low-level languages and explain the differences between them. Understand that ultimately all programming code written in high-level or assembly languages must be translated into machine code. Understand that machine code is expressed in binary	They should know that all programs must be converted to machine code before they can be executed. Students should understand the advantages and disadvantages of programming in high-level and assembly languages. Students should know the purpose of and be able to compare the different types of translator.		examples of assembly language and machine code is helpful. Students could be given some opportunity to write very simple programs in assembly language so that they can see how assembly language compares with a high-level language but this is not a requirement. Students using Visual Basic or C# could find the exe file that is output when they compile a program. Students could look at an assembly language instruction set (eg ARM) to consider the	Video comparing C code with equivalent assembly language code Video on translation software	

Specification reference	Specification content	Learning outcomes	Suggested timing (lessons)	Possible teaching and learning activities	Resource	Exam 'hints and tips'
	and is specific to a processor or family of processors. Understand the advantages and disadvantages of low-level language programming compared with high-level language programming. Understand that there are three common types of program translator: interpreter, compiler, assembler.			types of instruction available and their limitations.		

Specification reference	Specification content	Learning outcomes	Suggested timing (lessons)	Possible teaching and learning activities	Resource	Exam ‘hints and tips’
	Explain the main differences between these three types of translator and understand when it would be appropriate to use each type of translator.					

3.1, 3.2 and 3.8 Programming consolidation/Aspects of software development/Controlled assessment practice

Total teaching time: 20 hours

Once students have developed their programming skills it’s important they get the opportunity to consolidate them by working on other programming projects. These could be set by the teacher for the whole class or individually chosen to reflect students’ interests and ability.

When completing consolidation tasks, students need to develop their own skills in analysing problems and designing and testing their solutions, as well as coding them. These skills will be important for the completion of the controlled assessment task. The specification requires that, in relation to aspects of software design, students should cover these topics:

- **Design**: Be aware that before constructing a solution, the solution should be designed, for example planning data structures for the data model, designing algorithms, designing an appropriate modular structure for the solution and designing the user interface.
- **Implementation**: Be aware that the models and algorithms need to be implemented in the form of data structures and code (instructions) that a computer can understand.
- **Testing**: Be aware that the implementation must be tested for the presence of errors, using selected test data covering normal (typical), boundary (extreme) and erroneous data.
- **Evaluating/refining**: Be aware that code created during implementation will often require refining as a result of testing. Be aware of the importance of assessing how well the solution meets the requirements of the problem and how the solution could be improved if the problem were to be revisited.

Past and sample coursework (NEA) assignments set for GCSE coursework are one source of ideas for practising software development techniques and for consolidating programming tasks.

Students should complete exercises that require them to complete formal design, testing and evaluation work in addition to programing so that they are able to demonstrate these tasks in the NEA. They might also look at code that is provided to them and consider writing a test plan for it or evaluating it.

Able students could also be given the opportunity to extend their skills; for example, if a student learnt how to program in console mode they could be given the opportunity to develop applications with a graphical user interface.

3.3 Data representation

Total teaching time: 8.5 hours

The topics in this section of the specification all require students to apply their skills, so it’s important that they get plenty of opportunities to do this.

Specification reference	Specification content	Learning outcomes	Suggested timing (lessons)	Possible teaching and learning activities	Resource	Exam 'hints and tips'
3.3.1, 3.3.2	Explain why computers use binary. Understand how binary can be used to represent whole numbers and be able to convert between binary and decimal and vice-versa.	Understand that computers use binary to represent data and instructions. Be able to convert between binary and decimal and vice-versa.	1 hour	Look at how computers store data conceptually as on and off states and how this can be conceived numerically as binary (may be easier to look at early computers with valves, transistors). Review how the decimal system works with 10 digits and place values that are powers of 10 and relate this to how binary works with 2 digits and place values that are powers of 2. Show how a binary number can be converted to decimal by adding the place values of columns with 1s in.	Notes on binary and conversion Binary conversions game	Exam questions will only use values up to 8 bits in length.

Specification reference	Specification content	Learning outcomes	Suggested timing (lessons)	Possible teaching and learning activities	Resource	Exam 'hints and tips'
				Show how decimal can be converted to binary by working from left to right. Consider the highest and lowest decimal value that can be stored in 8 bits. This is a topic that students must practise, so they need to complete conversion exercises, possibly some in class and some for homework.		
3.3.1, 3.3.2	Understand how hexadecimal can be used to represent whole numbers and be able to convert between decimal and hexadecimal and	Be able to convert between decimal and hexadecimal and binary and hexadecimal. Understand why hexadecimal is often used in computer science and	1 hour	Consider why binary is not easy for humans to use (eg long strings of digits, easy to transpose, hard to remember). Explain why hexadecimal is a good shorthand for binary (4 bits = 1 hex digit) and look at where hex is used eg colour	Online notes on conversions	Exam questions will only use 8-bit values.

Specification reference	Specification content	Learning outcomes	Suggested timing (lessons)	Possible teaching and learning activities	Resource	Exam 'hints and tips'
	binary and hexadecimal. Understand why hexadecimal is often used in computer science.	give examples of where it is used.		codes, MAC addresses, memory editors. Look at methods for converting between decimal and hexadecimal and vice-versa (remember only 8-bit numbers are needed). Look at the quick method for converting between binary and hexadecimal and vice-versa in groups of 4 bits. Students needs to complete plenty of example conversion exercises in class and for homework.		
3.3.3	Know the units that are used to measure quantities of bytes.	Students know the units bit, byte, kilobyte,	0.5 hours	Explain the names of the measurements used for quantities. Consider a comparison with measurements		Students need to remember that for this specification, powers of 10 are

Specification reference	Specification content	Learning outcomes	Suggested timing (lessons)	Possible teaching and learning activities	Resource	Exam ‘hints and tips’
		megabyte, gigabyte and terabyte.		for distance where different but related measurements are used depending on the magnitude of the distance being measured (eg cm, m, km). Emphasise that this specification uses the SI definitions of the units which are powers of 10, but refer to the historical definitions using powers of 2, which students may be familiar with. Look at measurements of sizes of typical things eg RAM in a computer, size of a hard disk, download allowances. Students could be set some exercises working out file sizes or converting between units.		used for units not powers of 2.

Specification reference	Specification content	Learning outcomes	Suggested timing (lessons)	Possible teaching and learning activities	Resource	Exam 'hints and tips'
3.3.4	Be able to add together up to three binary numbers. Be able to perform logical shifts.	Be able to add together three binary numbers. Be able to perform logical shifts.	0.5 hours	Students need to be shown the method for completing binary addition of three numbers, including how to deal with multiple carries. Then they should complete some exercises to practice this. Students should then be shown how a binary shift can be used to double/ approximately halve a number.	Notes on binary addition (note that negative numbers are not required) Binary addition video	
3.3.5	Understand character sets including ASCII and Unicode and the advantages of Unicode.	Understand character sets including ASCII and Unicode and the advantages of Unicode. Be able to use a character table to convert a message from binary to a	1 hours	Look at the ASCII table. Complete exercise converting a message from binary to characters and vice-versa.	Online notes on character sets The ASCII table	Students need to be aware that similar characters run in blocks eg A = 65, B = 66 etc.

Specification reference	Specification content	Learning outcomes	Suggested timing (lessons)	Possible teaching and learning activities	Resource	Exam 'hints and tips'
		character set and vice-versa.		Note how similar characters are in blocks eg all capital letters. Consider limitations of ASCII (limited number of characters) and look at how Unicode solves these. This topic could be linked to programming through the use of the programming language commands for conversion between character codes and characters.	Official Unicode website Unicode character tables	
3.3.6	Understand how images can be represented as bitmaps, including key terms.	Define key terms eg pixel, colour depth. Calculate file sizes for bitmap images.	1.5 hours	Look at bitmap images using a graphics package, use zoom to identify pixels and colours (possible link to hex). Introduce colour depth by considering how different patterns of 0s and 1s could be	A bitmap image editor eg Paint. Online notes and test (note: vector graphics are not required)	

Specification reference	Specification content	Learning outcomes	Suggested timing (lessons)	Possible teaching and learning activities	Resource	Exam 'hints and tips'
	Be able to calculate file sizes. Be able to convert between binary and image data for black and white images.	Convert between an image and binary and vice-versa.		used to represent colours. A colour depth of n bits allows 2^n colours. Perform some exercises where students have to convert small images between images and binary data and vice-versa. Only black and white images need to be considered. Explain how to calculate the size of an image file and then students complete some sample calculations.		
3.3.7	Understand analogue sound must be converted to digital form for storage.	Understand the difference between analogue and digital.	1.5 hours	Discuss difference between analogue and digital quantities. Look at how sound can be represented electronically as a waveform – a package such as Audacity can be used to allow	A sound editing package such as Audacity Video (goes beyond GCSE level)	

Specification reference	Specification content	Learning outcomes	Suggested timing (lessons)	Possible teaching and learning activities	Resource	Exam ‘hints and tips’
	Describe how sound is represented using sample rate and sample resolution. Calculate sound file sizes.	Be able to recognise analogue and digital quantities. Be able to explain how sample rate and sample resolution affect sample quality and file size. Be able to calculate file sizes for sound files.		students to look at sounds and record their own. Use a graph to show how the sampling process works and how sample quality and size would be affected by changing sample rate and sample resolution. Perform calculations of sound file sizes. Students should carry out exercises that involve identifying analogue and digital quantities, converting between an analogue waveform and digital samples and calculating sound file sizes.		

Specification reference	Specification content	Learning outcomes	Suggested timing (lessons)	Possible teaching and learning activities	Resource	Exam 'hints and tips'
3.3.8	Explain what data compression is, and why it is used. Be able to explain how Huffman coding works and know how to use a tree to decompress data using Huffman coding and calculate how many bits are saved. Be able to compress/ decompress data using RLE.	Explain what data compression is, and why it is used. Be able to decompress data using Huffman coding and calculate how many bits are saved. Be able to compress/decompress data using RLE.	1.5 hours	Students could try creating ZIP files or comparing the size of JPEG (compressed) and Bitmap files of the same image to see the effect of compression. In discussion, consider why compression is useful – either in the context of transmission or storage of data eg faster downloads, more photos on memory cards etc. RLE is the simplest of the two techniques so it is best to cover this first. Look at how it can be used with small images and get students to try compressing small bitmaps using it. Consider why it is not suitable for some images and many types of data.	Compression methods and RLE video RLE video Huffman coding video	

Specification reference	Specification content	Learning outcomes	Suggested timing (lessons)	Possible teaching and learning activities	Resource	Exam 'hints and tips'
				Look at Huffman coding and the concept of variable-length codes with more common characters having shorter codes. Students should try to use a Huffman tree to decode some text stored as binary data. At this point, a calculation of how much memory was saved compared to using 7-bit ASCII can be made. Students do not need to be able to build a Huffman coding tree. Students need to complete practice exercises compressing and decompressing data using both RLE and Huffman coding and calculating how much		

Specification reference	Specification content	Learning outcomes	Suggested timing (lessons)	Possible teaching and learning activities	Resource	Exam 'hints and tips'
				memory is saved when Huffman coding is used. There are lots of videos available illustrating these techniques.		

3.5 Computer networks

Total teaching time: 4 hours

Many of the topics in this section are quite theoretical, so could be delivered through discussion and students using textbook/notes. It's important that students have the opportunity to demonstrate their understanding by answering questions. Some practical networking could be done using, for example, Raspberry Pi computers, which students can build a network from themselves, but this is not a requirement.

Specification reference	Specification content	Learning outcomes	Suggested timing (lessons)	Possible teaching and learning activities	Resource	Exam 'hints and tips'
3.5	Define what a computer network is. Discuss the benefits and risks of computer networks. Understand that networks can be wired or wireless. Discuss the benefits and risks of wireless networks as opposed to wired networks.	Students should be able to explain what a computer network is, discuss risks and benefits of networks and the relative merits of wired and wireless networking.	1 hour	Students will have direct experience of using networks, both wired and wireless, so this makes a good discussion topic – pros and cons of having a network and also of wired vs wireless networks. Devices such as Raspberry Pis could be used to build a network if it is desired that students have some practical experience.	Online notes and test (note also covers some topics that are not required)	
3.5	Describe the LAN, WAN and PAN types of computer network.	Students can describe LAN, WAN and PAN and understand star and bus topologies, including their relative merits.	0.5 hours	Differences between LAN and WAN should be considered in terms of size, ownership and the hardware used.	Online notes and test (note also covers some topics that are not required)	

	Explain the star and bus physical network topologies.			Topologies are best visualised; it is worth noting that physical bus networks have limited applications nowadays. This topic can be taught as a discussion or there are many online videos and resources.	Video (covers more topologies than needed):	
3.5	Define the term 'network protocol'. Explain the purpose and use of common network protocols including: Ethernet, Wi-Fi, TCP, UDP, IP, HTTP, HTTPS, FTP, SMTP, IMAP.	Students understand and can describe what a protocol is. Students can explain the purpose of the protocols and their use. They don't need to know any technical details about implementation.	1 hour	This topic is very theoretical and is probably best taught with students reading notes or the teacher delivering a presentation. Students should then answer questions that test their understanding. It is possible to demonstrate the use of some of the protocols, for example by using Telnet to open connections to a web server or email server, but this is not required for GCSE. Some online resources are also available.	Online notes (pages 5 and 6 have basic coverage of protocols)	

3.5	Understand the need for, and importance of, network security. Explain the following methods of network security: authentication, encryption, firewall, MAC address filtering.	Students understand why security is important on networks (more so than standalone computers) and the listed security measures.	0.5 hours	This topic can be taught theoretically or, if the teacher has access to this, students could be shown how some of these measures are used in school, for example firewall rules used.	Online notes and test Video showing use of MAC address whitelist Very short video on firewalls	
3.5	Describe the 4 layer TCP/IP model. Understand that the HTTP, HTTPS, SMTP, IMAP and FTP protocols operate at the application layer. Understand that the TCP and UDP protocols operate at the transport layer.	Students should know what the four layers are and some functions of each layer, together with which of the protocols listed operate at which layer.	1 hour	This topic is a fairly theoretical one. Students could use textbooks, online notes or videos to learn from. They need to understand why a stack is used (abstraction), what the four layers are and some functions of each layer of the stack and at which layers the listed protocols work.	Video	The specification uses the four layer TCP/IP model. There is an alternative five layer model and also a seven layer OSI model, neither of which are required.

	Understand that the IP protocol operates at the network layer.					

3.6 Cyber security

Total time for teaching this section: 2.5 hours

Specification reference	Specification content	Learning outcomes	Suggested timing (lessons)	Possible teaching and learning activities	Resource	Exam 'hints and tips'
3.6, 3.6.1, 3.6.1.1, 3.6.1.2	Define the term cyber security and be able to describe the main purposes of cyber security. Understand and be able to explain the following cyber security threats: social engineering	Be able to explain cyber security and the cyber security threats covered by the specification.	1.5 hours	This topic works well as a class discussion as most students will be familiar with some of these topics from their own personal experiences. Students could make a presentation, each focusing on one or more topics.	Lots of cyber security resources including lesson plans and games Documentary on cybercrime in the UK Five of the worst computer viruses	

Specification reference	Specification content	Learning outcomes	Suggested timing (lessons)	Possible teaching and learning activities	Resource	Exam 'hints and tips'
	techniques, malicious code, weak and default passwords, misconfigured access rights, removable media, unpatched and/or outdated software. Describe what social engineering is. Explain the following forms of social engineering: blagging, phishing, pharming, shouldering Define the term 'malware'.				Notes on some topics of computer security	

Specification reference	Specification content	Learning outcomes	Suggested timing (lessons)	Possible teaching and learning activities	Resource	Exam ‘hints and tips’
	Describe the following forms of malware: computer virus, Trojan, spyware, adware.					
3.6.1, 3.6.1.1, 3.6.1.2, 3.6.2	Describe how social engineering can be protected against. Describe how malware can be protected against. Understand and be able to explain the following security measures biometric measures, password systems, CAPTCHA, using email confirmations,	Be able to describe methods that are suitable for protecting from cyber security threats	1 hour	This topic woks well as a discussion, as students will be aware of some of these topics from their own experiences. They may need to be focused somewhat to ensure that they cover all of the topics on the specification. A range of useful online videos are available.	Novalabs cyber security protection game Cyber security threats and solutions	

Specification reference	Specification content	Learning outcomes	Suggested timing (lessons)	Possible teaching and learning activities	Resource	Exam 'hints and tips'
	automatic software updates. Explain what penetration testing is and what it is used for.					

3.7 Ethical, legal and environmental impacts including privacy

Total teaching time: 4 hours

This section of the specification is well suited to class discussions, debates with students taking opposing sides of an issue and students completing individual research and perhaps making presentations. Exam questions on this section will be drawn from the following areas:

- cyber security
- mobile technologies
- wireless networking
- cloud storage
- theft of computer code

- issues around copyright of algorithms
- cracking
- hacking
- wearable technologies
- computer based implants.

Throughout this section, students should be referred back to the need to consider these examples in the context of their ethical, legal and environmental impact on society (not all of these are relevant to each example).

Specification reference	Specification content	Learning outcomes	Suggested timing (lessons)	Possible teaching and learning activities	Resource	Exam 'hints and tips'
3.7	Hacking, cracking, cloud storage, cyber security	Students understand what hacking is, what the dangers of hacking are and how cyber security measures can be used to prevent it. Students understand what cloud storage is and its advantage and disadvantages in	1 hour	Students should consider what hacking is and the motivation for it. A look at some simple hacking techniques might be of interest but is not required. This topic should be linked into cyber security and the measures that can be taken to prevent hacking. The issue of hacking by governments and whether this	There are many videos on hacking on YouTube, for example: • 5 most dangerous hackers of all time • 10 biggest computer hacks of all time	

Specification reference	Specification content	Learning outcomes	Suggested timing (lessons)	Possible teaching and learning activities	Resource	Exam ‘hints and tips’
		comparison to local storage. Students understand that cracking can be used as an alternative name for hacking or to mean breaking copy protection of software.		can be justified and under what circumstances could also be looked at. The advantages and disadvantages of cloud storage should be considered. This would probably be best achieved in the context of a real cloud service that students’ might have used. Examples of security breaches of such services could be examined. Cyber security methods are considered in greater detail in specification section 3.6. This topic could be tackled by students doing individual research and then having a class discussion.	• Hacking a car with an ex-NSA hacker Cloud storage	

Specification reference	Specification content	Learning outcomes	Suggested timing (lessons)	Possible teaching and learning activities	Resource	Exam ‘hints and tips’
3.7	Copyright of algorithms, theft of computer code.	Students understand what intellectual property is and how it applies to computer algorithms and software and the potential effects of software code theft.	0.5 hours	Students should consider what intellectual property is and how this relates to algorithms and computer programs. They could look at laws covering copyright, but detailed knowledge of these is not required. The problem of theft of computer code and the impact of this on companies that write software and more widely should be considered. Different types of licensing of software could be looked at, but this is not required for the specification.	Government website on intellectual property BBC Bitesize	

Specification reference	Specification content	Learning outcomes	Suggested timing (lessons)	Possible teaching and learning activities	Resource	Exam 'hints and tips'
3.7	Wireless networking and mobile technologies.	Students should understand the risks and benefits of wireless and mobile technologies.	0.5 hours	Students should consider the risks and benefits of wireless networking, including ease of access to the Internet and possible security risks. Mobile technologies are facilitating many new applications of computing and also making access to the Internet more widely available. Some of this content may have already been covered as part of specification section 3.5. This topic would be suitable for students to do individual research on and then have a class discussion about.	Article on risks of wireless networks Network security video Video on mobile technology	

Specification reference	Specification content	Learning outcomes	Suggested timing (lessons)	Possible teaching and learning activities	Resource	Exam ‘hints and tips’
3.7	Wearable technology and computer based implants.	Students should be familiar with applications of wearable technologies and computer based implants, and aware of the risks associated with them.	1 hour	Students should be aware of applications of wearable technology and computer based implants. They should also consider the risks of these technologies. Students could research applications and risks and put together presentations either individually or with students contributing towards a group presentation.	Examples of implants Downloading into brains (video) TED talks on wearable technologies	
3.7	Role of the state – privacy vs safety.	Students should understand that privacy and the need for the state to protect its citizens can come into conflict.	1 hour	Students need to consider the balance between privacy for individuals and the need for the state to protect people from crime and terrorism. This topic has arisen many times recently in the news. For	Resources from the UK government	

Specification reference	Specification content	Learning outcomes	Suggested timing (lessons)	Possible teaching and learning activities	Resource	Exam 'hints and tips'
		They should be able to make arguments on either side of this case and be able to support their arguments with examples.		example, Edward Snowden/WikiLeaks and use of encrypted messaging systems by terrorists. These would form useful case studies where students could consider the relative merits of the arguments on either side of the debate.		

Assessment and exam preparation

Total teaching time: 5 hours

It's important that students are formally assessed on both their programming skills and their theoretical knowledge.

Completing programming tasks under timed conditions and without teacher assistance will help the teacher to identify students who are finding the work challenging so that they can intervene early to help.