Key Stage 4 Academic Computing Curriculum
AQA GCSE Computer Science

National Curriculum Key stage 4
All pupils must have the opportunity to study aspects of information technology and computer science at sufficient depth to allow them to progress to higher levels of study or to a professional career.
All pupils should be taught to:

- develop their capability, creativity and knowledge in computer science, digital media and information technology
- develop and apply their analytic, problem-solving, design, and computational thinking skills
- understand how changes in technology affect safety, including new ways to protect their online privacy and identity, and how to identify and report a range of concerns.

AQA GCSE Computer Science Aims and learning outcomes

- Courses based on this specification must encourage students to:
- build on their knowledge, understanding and skills established through the computer science elements of the programme of study for computing at Key Stage 3 and Key Stage 4
- enable students to progress into further learning and/or employment
- understand and apply the fundamental principles and concepts of computer science, including abstraction, decomposition, logic, algorithms, and data representation
- analyse problems in computational terms through practical experience of solving such problems, including designing, writing and debugging programs
- think creatively, innovatively, analytically, logically and critically
- understand the components that make up digital systems, and how they communicate with one another and with other systems
- understand the impacts of digital technology to the individual and to wider society
- apply maths skills relevant to computer science.

Assessment objectives
AO1: Demonstrate knowledge and understanding of the key concepts and principles of computer science.
AO2: Apply knowledge and understanding of key concepts and principles of computer science.
AO3: Analyse problems in computational terms: to make reasoned judgements to design, program, evaluate and refine solutions.

Key Stage 4 Academic Computing Curriculum
AQA GCSE Computer Science

Year 10 Academic	Autumn 1	Autumn 2	Spring 1	Spring 2	Summer 1	Summer 2
Topic	**3.1 Fundamentals of algorithms and 3.2 Programming**	**3.1 Fundamentals of algorithms and 3.2 Programming**	**3.1 Fundamentals of algorithms and 3.2 Programming**	**3.1 and 3.2 Programming consolidation**	**3.1 and 3.2 Programming consolidation**	**3.3 Fundamentals of data representation**
Knowledge: Pupils will learn how to:	**Specification reference 3.1.1, 3.2.1, 3.2.2, 3.2.3, 3.2.7** Understand and explain the term algorithm. Understand and use string, integer and real data types appropriately. Understand how variable declaration and assignment can be used in programs. Use addition, subtraction, multiplication and real division. Output data and information from a program to a computer display. Use meaningful identifier names and know why it's important to use them. **Specification reference 3.2.2, 3.2.4, 3.2.5, 3.2.11** Use selection (if, else, else if, case/switch if appropriate) Use a range of relational operators ie equal to, not equal to, less than, greater than, less than or equal to and greater than or equal to.it Use NOT, AND, OR. Using nested selection structures. Understand what is meant by testing and be able to correct errors in programs and algorithms.	**Specification reference 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. 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. **Specification reference 3.1.1, 3.2.2, 3.2.3, 3.2.10**	**Specification reference 3.22, 3.2.6** Use two-dimensional arrays (or equivalent) in the design of solutions to simple problems. Use nested iteration. Use of constants. **Specification reference 3.2.6, 3.2.1** Use records (or equivalent) in the design of solutions to simple problems. Be able to write simple authentication routines **Specification reference 3.1.1** Use a systematic approach to problem solving and algorithm creation representing those algorithms using pseudo-code, program code and flowcharts. Explain simple algorithms in terms of their inputs, processing and outputs. Determine the purpose of simple algorithms.	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. While there is no longer coursework or non-exam assessment (NEA) programming tasks, by working on larger projects, students are maturing and embedding their programming skills for Paper 1.	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. While there is no longer coursework or non-exam assessment (NEA) programming tasks, by working on larger projects, students are maturing and embedding their programming skills for Paper 1.	**Specification reference 3.3.1, 3.3.2** Understand the number bases decimal (base 10) and binary (base 2). Understand that computers use binary to represent all data and instructions. Understand how binary can be used to represent whole numbers and be able to convert between binary and decimal and vice-versa. **Specification reference 3.3.1, 3.3.2** Understand the number base hexadecimal (base 16). Understand how hexadecimal can be used to represent whole numbers and be able to convert between decimal and hexadecimal as well as binary and hexadecimal. Understand why hexadecimal is often

Key Stage 4 Academic Computing Curriculum
AQA GCSE Computer Science

	Select suitable test data that covers normal (typical), boundary and erroneous data. Be able to justify the choice of test data. Understand pseudo-code and flowcharts. Understand that there are different types of error including syntax and logical errors. Identify and categorise errors in algorithms and programs. **Specification reference 3.1.1, 3.2.2, 3.2.8, 3.2.9, 3.2.11** Use indefinite iteration with conditions at start and end of loop. Use random number generation. Use some string handling techniques: ● length ● position ● substring ● concatenation convert character to character code convert character code to character string conversion operations. Write simple data validation routines. Write simple authentication routines. Understand and explain the term abstraction. **Specification reference 3.2.2** Use definite iteration. Use nested iteration.	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. Explain the advantages of using subroutines in programs. Integer division, including remainders. **Specification reference 3.2.1, 3.2.8, 3.2.10** Use a structured approach to programming, in particular focussing on the use of parameters and return values. Use a range of string handling operations from: *length *position *substring *concatenation *convert character to character code *convert character code to character				used in computer science. **Specification reference 3.3.3** Know the units that are used to measure quantities of bytes. **Specification reference 3.3.4** Add together up to three binary numbers. Perform logical shifts. Describe situations where binary shifts can be used. **Specification reference 3.3.5** Understand character sets including ASCII and Unicode and the advantages of Unicode. Understand that character codes are commonly grouped and run in sequence within encoding tables. **Specification reference 3.3.6** Understand how images can be represented as bitmaps, including key terms. Calculate file sizes. Convert between binary and image data for simple images. **Specification reference 3.3.7** Understand analogue sound must be sampled and

Key Stage 4 Academic Computing Curriculum
AQA GCSE Computer Science

		*string conversion operations. Use the char and Boolean data types.				converted to digital form for storage and processing. Describe how sound is represented using sample rate and sample resolution. Calculate sound file sizes. **Specification reference 3.3.8** Explain what data compression is, why it's used and why there are different methods used. Explain how Huffman coding works and know how to use a tree to decompress data using Huffman coding and calculate how many bits are used vs how many bits are stored using uncompressed ASCII data. Compress/ decompress data using RLE.
Skills	**Specification reference 3.1.1, 3.2.1, 3.2.2, 3.2.3, 3.2.7** Introduce students 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 e.g. how to run a program, how to load/save,	**Specification reference 3.1.1, 3.1.2, 3.1.3, 3.1.4, 3.2.6** Introduce students to the concept of a one-dimensional array and give them the opportunity to solve problems using them. Cover the four searching and sorting	**Specification reference 3.22, 3.2.6** Give students the opportunity to write programs using two-dimensional arrays. They'll need to consider/design how the arrays can be used to represent the problem. Data stored in a two-dimensional array is usually displayed most	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. The more problems that are solved the better – programming the	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. The more problems that are solved the better – programming the solutions to the problems just allows students to check their solutions work. The	**Specification reference 3.3.1, 3.3.2** Look at how computers store data conceptually as high/low voltage and on and off states and how this can be conceived numerically as binary (may be easier to look at early

Key Stage 4 Academic Computing Curriculum
AQA GCSE Computer Science

	how error messages are presented and what they mean. Introduce students to the idea of an algorithm and that a program is an implementation of an 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, eg adding three numbers, multiplying two numbers together. This could be done in 'shell' mode but will be better contextualised if made into the form of a full program. *doing more complex calculations, e.g. area of a rectangle, area of a triangle, area of a circle, area of a trapezium. Students could be set the task of completing problems from their maths classes as programs. *students converting between provided code, pseudocode and flowcharts. **Specification reference 3.2.2, 3.2.4, 3.2.5, 3.2.11** Teach pupils about the use of selection statements to determine the path of code execution. Exercises should build in difficulty, starting with	algorithms and give the opportunity to code them, except the merge sort. Before coding these algorithms, it'd 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. It isn't expected that concepts such as Big-O or T(n) are introduced. However, counting operations can help demonstrate efficiency. Exercises could include: *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	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 • battleships. Python programmers should use lists rather than importing Array classes for simplicity. **Specification reference 3.2.6, 3.2.1** Introduce students 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 *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) *adapt the adventure game from earlier to store	solutions to the problems just allows students to check their solutions work. The specification requires that, for Paper 1, students should have sufficient practice of: • structuring programs into modular parts with clear documented interfaces to enable them to design appropriate modular structures for solutions • including authentication and data validation systems/routines within their computer programs • writing, debugging and testing programs to enable them to develop the skills to articulate how programs work and argue using logical reasoning for the correctness of programs in solving specified problems • designing and applying test data (normal, boundary and erroneous) to the testing of programs so that they are familiar with these test data	specification requires that, for Paper 1, students should have sufficient practice of: • structuring programs into modular parts with clear documented interfaces to enable them to design appropriate modular structures for solutions • including authentication and data validation systems/routines within their computer programs • writing, debugging and testing programs to enable them to develop the skills to articulate how programs work and argue using logical reasoning for the correctness of programs in solving specified problems • designing and applying test data (normal, boundary and erroneous) to the testing of programs so that they are familiar with these test data types and the purpose of testing • refining programs in response to testing outcomes. Past and sample coursework assignments set for GCSE coursework are one source of ideas for practising and consolidating programming tasks, as are the programming challenges available on our website. Able students could also be given the opportunity to extend their skills; for	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. 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. **Specification reference 3.3.1, 3.3.2** 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

Key Stage 4 Academic Computing Curriculum
AQA GCSE Computer Science

	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. Use pseudocode and flowcharts to illustrate some algorithms which students could then write program code for. Whilst completing these exercises, consider choosing test data, which is particularly important in boundary situations. Introduce deliberate errors (a) to introduce students to syntax errors and (b) to demonstrate how logical errors may only be picked up by thorough testing. **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. *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 zero0C or below = freezing, above zero0C but 100C or below = warm. *Find the error activities. **Specification reference 3.1.1, 3.2.2, 3.2.8, 3.2.9, 3.2.11**	(eg names) in an array *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 3.1.1, 3.2.2, 3.2.3, 3.2.10** Teach students about why, when writing longer programs, it's 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	each room as a record within an array. **Specification reference 3.1.1** Throughout learning to program, expose students to how algorithms can be expressed using pseudo-code or flowcharts. 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. This may include use of records, string functions and two-dimensional arrays. These skills will be assessed in the exam. It's useful to teach them in parallel with learning to program (perhaps as homework exercises) but it could also be worth giving students the opportunity to consolidate their ability to apply these skills. Students should complete exercises where they have to read and write pseudocode and flowcharts, complete trace tables and deduce the purpose of algorithms.	types and the purpose of testing • refining programs in response to testing outcomes. Past and sample coursework assignments set for GCSE coursework are one source of ideas for practising and consolidating programming tasks, as are the programming challenges available on our website. 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. Problems from Paper 1 of the AQA AS/A-level exams may also be used to stretch more able students.	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. Problems from Paper 1 of the AQA AS/A-level exams may also be used to stretch more able students.	bits = 1 hex digit) and look at where hex is used eg colour 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 need to complete plenty of example conversion exercises in class and for homework. It's worth considering whether your students would learn a direct decimalhexadecimald ecimal conversion method or would be better using binary as an intermediary. **Specification reference 3.3.3** Explain the names of the measurements used for quantities. Consider a comparison with measurements for distance where different but related measurements are used depending on the magnitude of the distance being

Key Stage 4 Academic Computing Curriculum
AQA GCSE Computer Science

	Students need to know about indefinite iteration and how to use this in their programming language. For students using Python, which does not have a post-conditioned loop, you should teach 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, i.e. 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. This can include using the LENGTH string handling technique *adding up a sequence of numbers of unknown length *asking users to enter a password until the correct password is entered, displaying suitable messages *guessing randomly chosen number until they guess correctly, with clues given about whether guess is too high/low	that's used to call different subroutines to work out (for example) the area of different shapes *making a program that'll allow conversion of numbers between different number bases, with different functions being used for different conversions eg binary to decimal *creating an adventure game with different rooms/actions having different subroutines. In all subsequent programs, encourage students to consider how the programs can be decomposed into subroutines. **Specification reference 3.2.1, 3.2.8, 3.2.10** 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 keyboard/screen should not happen within the functions. Teach students why this is important, eg in terms of being able to develop and test				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. You could set students some exercises working out file sizes or converting between units. Give students exercises for comparing between prefixes – eg what is larger, 3,000 Megabytes or 4 Gigabytes? **Specification reference 3.3.4** Show students the method for completing binary addition of three numbers, including how to deal with multiple carries. Then they should complete some exercises to practise this. Then show students how a binary shift can be used to double/

Key Stage 4 Academic Computing Curriculum
AQA GCSE Computer Science

	*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 zero is achieved). **Specification reference 3.2.2** Introduce students to the concept of definite iteration and a loop counter. Use pseudocode and flowcharts to illustrate algorithms. **Exercises could include:** *counting from one to 10 *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.	modules independently and reuse code. Returning tuples (in Python) should be discouraged as it can lead to language-specific pseudo-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 *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 (using string position, and conversion between character codes) *developing a function to convert a string into Morse code *developing a function that will return				approximately halve a number. **Specification reference 3.3.5** Look at the ASCII table. Complete exercise converting a message from binary to characters 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. **Specification reference 3.3.6** 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 used to represent colours. A colour depth of n bits allows 2n colours.

Key Stage 4 Academic Computing Curriculum
AQA GCSE Computer Science

		a true/false value, 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, encourage students to consider how the programs can be decomposed into functions with interfaces that use parameters and return values.				Perform some exercises where students have to convert small images between images and binary data and vice-versa. Explain how to calculate the size of an image file and then students complete some sample calculations. **Specification reference 3.3.7** 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 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. **Specification reference 3.3.8** Students should carry out exercises that involve identifying analogue and digital quantities, converting between an analogue

Key Stage 4 Academic Computing Curriculum
AQA GCSE Computer Science

						waveform and digital samples and calculating sound file sizes. 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's 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 isn't suitable for some images and many types of data. Look at Huffman coding and the concept of variable-length codes with more common characters having shorter codes. Students should try using a Huffman tree to decode some text stored as binary data. At this point, a

Key Stage 4 Academic Computing Curriculum
AQA GCSE Computer Science

						calculation of how much memory was saved compared to using 7-bit ASCII can be made. Students don't need to be able to build a Huffman coding tree although doing so can aid understanding. Students need to complete practice exercises compressing and decompressing data using both RLE and Huffman coding and calculating how much memory is saved when Huffman coding is used. There are lots of videos available illustrating these techniques.
Vocabulary	All vocabulary needed for GCSE computer science can be found in this document - pupils should be familiar with all of these vocabulary terms: Subject Specific Vocabulary.PDF					

Key Stage 4 Academic Computing Curriculum
AQA GCSE Computer Science

Year 11 Academic	Autumn 1	Autumn 2	Spring 1	Spring 2	Summer 1	Summer 2
Topic	**3.4 Computer systems**	**3.5 Fundamentals of computer networks and 3.6 Cyber security**	**3.7 Relational databases and structured query language**	**3.8 Ethical, legal and environmental impacts including privacy**	**Assessment and exam preparation**	**Assessment and exam preparation**
Knowledge	**Specification reference 3.4.1, 3.4.3** Define the terms hardware and software and understand the relationship between them. Explain what's meant by system software and application software and be able to give examples of them. Understand the need for and functions of the OS and utility programs. **Specification reference 3.4.2** Construct truth tables for NOT, AND, OR, XOR gates, Construct truth tables for simple logic circuits and interpret them. Create, modify and interpret simple logic circuit diagrams. Create and interpret simple Boolean expressions made up of NOT, AND, OR and XOR operations. Convert between Boolean expression and logic circuits. **Specification reference 3.4.4**	**Specification reference 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. **Specification reference 3.5** Describe the LAN, WAN and PAN types of computer network. Explain the star and bus physical network topologies. **Specification reference 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. **Specification reference 3.5**	**Specification reference 3.7.1** Explain the concept of a database. Explain the concept of a relational database. Understand the following database concepts: o table o record o field o primary key o foreign key. Understand that the use of a relational database facilitates the elimination of data inconsistency and data redundancy. **Specification reference 3.7.2** Be able to use SQL to retrieve data from a relational database, using the commands:	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 • hacking (unauthorised access to a computer system) • wearable technologies • computer based implants • Autonomous vehicles. Throughout this section, students should be referred back to the need to consider these examples in the context of their ethical, legal	It's important that students are formally assessed on both their programming skills and their theoretical knowledge. Identified weaknesses will be retaught and tricky areas revised.	It's important that students are formally assessed on both their programming skills and their theoretical knowledge. Identified weaknesses will be retaught and tricky areas revised.

Key Stage 4 Academic Computing Curriculum
AQA GCSE Computer Science

	Know and explain the differences between low-level and high-level languages. Explain the differences between low-level languages (assembly language and machine code). Understand the need to translate high-level or assembly languages. Understand that machine code is expressed in binary and is specific to a processor or family of processors. Understand the advantages/ disadvantages of low-level vs high-level language programming. Understand and explain the differences between, and times to use, interpreters, compilers and assemblers. **Specification reference 3.4.5** Explain role of main memory, components of CPU, within the Von Neumann architecture: o arithmetic logic unit o control unit o clock o register o bus. Explain the effect of clock speed, number of cores and cache size on performance of the CPU. Understand and explain the fetch-execute cycle. **Specification reference 3.4.5** Understand the difference between main memory and secondary storage and between RAM, ROM, cache and a	Understand the need for, and importance of, network security. Explain the following methods of network security: authentication, encryption, firewall, MAC address filtering. **Specification reference 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. Understand that the IP protocol operates at the network layer. **Specification reference 3.6.1, 3.6.2** 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 techniques *malicious code *weak and default passwords *pharming *misconfigured access rights *removable media *unpatched and/or outdated software. Explain what penetration testing is and what it is used for. **Specification reference 3.6.2.1**	*SELECT *FROM *WHERE *ORDER BY..ASC | DESC Be able to use SQL to insert data into a relational database using the command: *INSERT INTO table_name (column1, column2 …) VALUES (value1, value2, …) Be able to use SQL to edit and delete data in a database using the commands: *UPDATE table_name SET column1 = value1, column2 = value2 … WHERE condition DELETE FROM table_name WHERE condition.	and environmental impact on society (not all of these are relevant to each example). **Specification reference 3.8** Explain the current ethical, legal and environmental impacts and risks of digital technology on society. Where data privacy issues arise, these should be considered.		

Key Stage 4 Academic Computing Curriculum
AQA GCSE Computer Science

	register, what they're used for and why they're required. Be aware of why secondary storage is needed and the 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. **Specification reference 3.4.4** Understand the term 'embedded system' and explain how an embedded system differs from a non-embedded system.	Describe what social engineering is. Explain the following forms of social engineering: blagging, phishing, shouldering Describe how social engineering can be protected against. **Specification reference 3.6.2.2** Define the term malware. Describe how malware can be protected against. Describe the following forms of malware: computer virus, Trojan, spyware. **Specification reference 3.6.3** Understand and be able to explain the following security measures biometric measures: *password systems *CAPTCHA *using email confirmations *automatic software updates.				
Skills	**Specification reference 3.4.1, 3.4.3** 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 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	**Specification reference 3.5** 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's desired that students have some practical experience. **Specification reference 3.5**	**Specification reference 3.7.1** This is a primarily theoretical section. However, the teacher could demonstrate the concepts by logging into an online SQL database. Data consistency/redundancy can be modelled on a student database and	**Specification reference 3.8** Students should be aware of applications and concepts of the technologies identified. Students could research applications and risks and put together presentations either individually or with students contributing towards a group presentation. Students should be made aware of ethical, legal and environmental impacts and what these mean.	Completing programming tasks under timed conditions and without teacher assistance will help to identify students who are finding the work challenging and prepare targeted revision lessons. They should also be explicitly taught exam techniques and literacy skills.	Completing programming tasks under timed conditions and without teacher assistance will help to identify students who are finding the work challenging and prepare targeted revision lessons. They should also be explicitly taught exam techniques and literacy skills.

Key Stage 4 Academic Computing Curriculum
AQA GCSE Computer Science

	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. Students need to know/understand that the OS manages processor(s), memory, I/O devices, applications and security but don't need to know how. A utility is a program that helps manage a computer but isn't 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. More able students might enjoy researching machines such as the PDP/8 which didn't have an OS and consider how such machines worked. Practice tasks arranging software into categories can be useful. **Specification reference 3.4.2** Consider the basic operations of AND, OR, XOR 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.	Differences between LAN and WAN should be considered in terms of size, ownership and the hardware used. Topologies are best visualised; it's 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. Students can build a bus and a star network out of string and coat hangers and demonstrate problems by cutting connections. **Specification reference 3.5** 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's 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 isn't required for GCSE. Students could carry out short research tasks to identify core reasons of the protocols. **Specification reference 3.5** 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, eg firewall rules used.	having a student "move to a new house", having parents marry, and so on. **Specification reference 3.7.2** This should be a practical activity with students given access to a real SQL database. There are online sites available or the teacher may set up a short-term hosted SQL server for the period of teaching this topic. Microsoft Access is unlikely to be suitable for teaching this topic. Students should be given the opportunity to run live SQL commands on a variety of sample data. Previous specification examinations and those from AQA AS/A-level papers may form suitable starting points for data sets and problems to explore.	Students should have practice writing and feeding back on written arguments for and against each of the topics identified in relation to their ethical, legal and environmental impacts as well as their privacy considerations. Resource: There are many videos on hacking on YouTube, for example: 5 most dangerous hackers of all time 10 biggest computer hacks of all time Resources from the UK government Article on risks of wireless networks Video on mobile technology Examples of implants Downloading into brains (video) TED talks on wearable technologies	Study and revision skills will be explicitly taught.	Study and revision skills will be explicitly taught.

Key Stage 4 Academic Computing Curriculum
AQA GCSE Computer Science

	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 problems. This could be done on paper, using an online logic circuit simulator or physically using electronics or tools such as logic goats. It isn't required for the specification but it'd 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. XOR can be demonstrated using a mask to change between upper- and lower-case ASCII. **Specification reference 3.4.4** This is a largely theoretical lesson which is most likely to be teacher-led. Students could research the subject matter and produce comparison charts. Mini case studies could be used to identify the most appropriate solutions and to engender discussion. **Specification reference 3.4.5** 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.	Roleplay can also be a useful tool with some students playing data packets and some students playing the security tool. **Specification reference 3.5** This topic is fairly theoretical. 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. **Specification reference 3.6.1, 3.6.2** 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. **Specification reference 3.6.2.1** This is an excellent opportunity for students to discuss personal experiences. Teachers may have examples of phishing attempts. The lesson could be started as simply as asking every student to write down their password and give it to their friend. The responses of the students, whether they comply or argue the point, can spark an interesting debate about personal security and what "social				

Key Stage 4 Academic Computing Curriculum
AQA GCSE Computer Science

	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 3.4.5** This isn't 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's helpful to focus on their uses. It's useful to have physical devices for students to look at here – a disassembled hard disk drive and CD-ROM drive or similar. There's less of interest that can be seen inside a solid state drive. 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.	engineering" can mean on a personal level. Ask any students who did write their passwords down to change them. **Specification reference 3.6.2.2** This topic works 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. **Specification reference 3.6.3** Most of these methods should be demonstrable within the classroom. Students should be encouraged to discuss the advantages and disadvantages of each and identify where and when each might be most appropriate.				

Key Stage 4 Academic Computing Curriculum
AQA GCSE Computer Science

	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've seen themselves but also the risks. **Specification reference 3.4.4** 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. Give students some scenarios (eg washing machine) and ask them to consider what functionality the system would need and why a non-embedded system wouldn't be suitable. Differences such as processor speed, amount and type of main memory, secondary storage, input and output devices and upgradeability could be considered. More able students could investigate the part embedded systems play within the Internet of Things.					
Vocabulary	All vocabulary needed for GCSE computer science can be found in this document - pupils should be familiar with all of these vocabulary terms: Subject Specific Vocabulary.PDF					