GCSE COMPUTER SCIENCE

(8525)

Specification

For teaching from September 2020 onwards
For GCSE exams in 2022 onwards

Version 1.2 29 November 2022

Contents

Are you using the latest version of this specification?

- You will always find the most up-to-date version of this specification on our website at
- We will write to you if there are significant changes to the specification.

1 Introduction

1.1 Why choose AQA for GCSE Computer Science

We've worked closely with teachers to develop a GCSE Computer Science specification that's as inspiring to teach as it is to learn. This specification recognises the well-established methodologies of computing, alongside the technological advances which make it such a dynamic subject.

We've built on the most popular aspects of our current specification and added fresh features including a programming exam to provide a programme of study for students of all ability levels. You can choose from a range of programming languages, enabling you to tailor your teaching to the strengths and preferences of yourself and your students.

Our exam papers retain our commitment to clear wording and structure, helping students to progress through each paper with confidence.

Students will complete this course equipped with the logical and computational skills necessary to succeed at A-level, the workplace or beyond.

As part of our ongoing commitment to provide excellent support, you'll see we've created fantastic free teaching resources and can offer great value professional development courses. We're also collaborating with publishers to ensure you have engaging and easy-to-use textbooks.

You can find out about all our Computer Science qualifications at aqa.org.uk/computer-science

1.2 Support and resources to help you teach

We've worked with experienced teachers to provide you with a range of resources that will help you confidently plan, teach and prepare for exams.

1.2.1 Teaching resources

Visit aqa.org.uk/8525 to see all our teaching resources. They include:

- sample papers and mark schemes to show the standards required and how your students' papers will be marked
- schemes of work and lesson plans to help you plan your course with confidence
- easy-to-use, AQA approved textbooks
- phone and email based subject team to support you in the delivery of the specification
- excellent professional development opportunities for those just starting out or the more experienced, looking for fresh inspiration
- training courses to help you deliver our computer science qualifications.

Preparing for exams

Visit aqa.org.uk/8525 for everything you need to prepare for our exams, including:

- sample papers and mark schemes for new courses
- Exampro: a searchable bank of past AQA exam questions
- example student answers with examiner commentaries.

Analyse your students' results with Enhanced Results Analysis (ERA)

Find out which questions were the most challenging, how the results compare to previous years and where your students need to improve. ERA, our free online results analysis tool, will help you see where to focus your teaching. Register at aqa.org.uk/era

For information about results, including maintaining standards over time, grade boundaries and our post-results services, visit aqa.org.uk/results

Keep your skills up-to-date with professional development

Time in the classroom is precious, but we understand that as your career develops, the skills and knowledge you need may change. As well as subject-specific training, we offer a range of courses to help boost your skills.

- Feedback sessions to understand how students have performed in this year's exams.
- Virtual communities where you can chat to teachers in your subject area about what's important to you.

This is just the start . Our events calendar is full of different ways for you to learn, understand and be inspired. Find out more at aqa.org.uk/professional-development.

Help and support

Visit our website for information, guidance, support and resources at aqa.org.uk/8525

If you'd like us to share news and information about this qualification, sign up for emails and updates at aqa.org.uk/from-2017

Alternatively, you can call or email our subject team direct.

E: computerscience@aqa.org.uk

T: 0161 957 3980

2 Specification at a glance

This qualification is linear. Linear means that students will sit all their exams at the end of the course.

2.1 Subject content

- 3.1 Fundamentals of algorithms (page 10)
- 3.2 Programming (page 11)
- 3.3 Fundamentals of data representation (page 18)
- 3.4 Computer systems (page 22)
- 3.5 Fundamentals of computer networks (page 27)
- 3.6 Cyber security (page 29)
- 3.7 Relational databases and structured query language (SQL) (page 31)
- 3.8 Ethical, legal and environmental impacts of digital technology on wider society, including issues of privacy (page 33)

2.2 Assessments

Paper 1: Computational thinking and programming skills
What's assessed
Computational thinking, code tracing, problem-solving, programming concepts including the design of effective algorithms and the designing, writing, testing and refining of code.
The content for this assessment will be drawn from subject content 3.1 and 3.2 above.
How it's assessed
• Written exam: 2 hours • 90 marks • 50% of GCSE
Questions
A mix of multiple choice, short answer and longer answer questions assessing programming, practical problem-solving and computational thinking skills.

Paper 2: Computing concepts
What's assessed
The content for this assessment will be drawn from subject content 3.3 to 3.8 above.
How it's assessed • Written exam: 1 hour 45 minutes • 90 marks • 50% of GCSE
Questions A mix of multiple choice, short answer, longer answer and extended response questions assessing SQL programming skills and theoretical knowledge.

3 Subject content

This subject content should be taught within a range of realistic contexts based around the major themes in the specification. To gain the most from the specification, a number of the sections will benefit from being taught holistically. For example, algorithms could be taught alongside programming techniques as there is a close relationship between them.

The specification content in Sections 3.1–3.8 is presented in a two-column format. The left hand column contains the specification content that all students must cover and that is assessed in the written papers. The right hand column exemplifies the additional information that teachers will require to ensure that their students study the topic in an appropriate depth and, where appropriate, gives teachers the parameters in which the subject will be assessed.

The skills, knowledge and understanding from all of the subject content within the specification will be assumed in all assessments. Both assessments may contain synoptic questions that will require students to use their skills, knowledge and understanding from across the entire specification. For example, whilst the understanding of binary numbers will be directly assessed in paper 2, the underlying knowledge and principles may be indirectly required for questions in paper 1.

For exams from 2022, we will support the following programming languages:

- C#
- Python (version 3)
- VB.NET.

In paper 1 students will be required to design, write, test and refine program code in one of the three languages above.

In preparation for paper 1, students should have sufficient practical experience 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.

In preparation for paper 2 students should have sufficient practical experience of writing and refining SQL.

Students should be given as much opportunity as possible to practise their programming skills in the school or college's chosen language and SQL.

3.1 Fundamentals of algorithms

3.1.1 Representing algorithms

Content	Additional information
Understand and explain the term algorithm.	An algorithm is a sequence of steps that can be followed to complete a task. Be aware that a computer program is an implementation of an algorithm and that an algorithm is not a computer program.
Understand and explain the term decomposition.	Decomposition means breaking a problem into a number of sub-problems, so that each sub-problem accomplishes an identifiable task, which might itself be further subdivided.
Understand and explain the term abstraction.	Abstraction is the process of removing unnecessary detail from a problem.
Use a systematic approach to problem solving and algorithm creation representing those algorithms using pseudo-code, program code and flowcharts.	Any exam question where students are given pseudo-code will use the AQA standard version. Exam questions will indicate the form of response expected. For example, pseudo-code, program code or a flowchart.
Explain simple algorithms in terms of their inputs, processing and outputs.	Students must be able to identify where inputs, processing and outputs are taking place within an algorithm.
Determine the purpose of simple algorithms.	Students should be able to use trace tables and visual inspection to determine how simple algorithms work and what their purpose is.

3.1.2 Efficiency of algorithms

Content	Additional information
Understand that more than one algorithm can be used to solve the same problem.	
Compare the efficiency of algorithms explaining how some algorithms are more efficient than others in solving the same problem.	Formal comparisons of algorithmic efficiency are not required. Exam questions in this area will only refer to time efficiency.

3.1.3 Searching algorithms

Content	Additional information
Understand and explain how the linear search algorithm works.	Students should know the mechanics of the algorithm.
Understand and explain how the binary search algorithm works.	Students should know the mechanics of the algorithm.
Compare and contrast linear and binary search algorithms.	Students should know the advantages and disadvantages of both algorithms.

3.1.4 Sorting algorithms

Content	Additional information
Understand and explain how the merge sort algorithm works.	Students should know the mechanics of the algorithm.
Understand and explain how the bubble sort algorithm works.	Students should know the mechanics of the algorithm.
Compare and contrast merge sort and bubble sort algorithms.	Students should know the advantages and disadvantages of both algorithms.

3.2 Programming

Students need a **theoretical understanding** of all the topics in this section for the paper 1 exam even if the programming language(s) taught does not support all of the topics. Exams will always present algorithms using the current version of the AQA pseudo-code. The document can be found on the AQA website.

Students need a **practical understanding** of all the topics and skills in this section for the paper 1 exam. When they are writing computer programs in an exam they will need to ensure they use meaningful identifier names, use suitable data types and that all logic flow is clearly identifiable to examiners.

Exam questions will explicitly state in what form the response needs to be provided. This will be, for example, pseudo-code, program code or a flowchart, and students must respond as instructed. Where pseudo-code is an accepted method of response, students may present their answers to questions in any suitable format and do not need to use the AQA pseudo-code.

3.2.1 Data types

Content	Additional information
Understand the concept of a data type.	

Content	Additional information
Understand and use the following appropriately: • integer • real • Boolean • character • string.	Depending on the actual programming language(s) being used, these data types may have other names. For example real numbers may be described as float. In exams we will use the general names given opposite.

3.2.2 Programming concepts

Content	Additional information
Use, understand and know how the following statement types can be combined in programs: • variable declaration • constant declaration • assignment • iteration • selection • subroutine (procedure/function).	The three combining principles (sequence, iteration/repetition and selection/choice) are basic to all high-level imperative programming languages. Students should be able to write programs using these statement types. They should be able to interpret and write algorithms that include these statement types. Students should know why named constants and variables are used.

Content	Additional information
Use definite (count controlled) and indefinite (condition controlled) iteration, including indefinite iteration with the condition(s) at the start or the end of the iterative structure.	A theoretical understanding of condition(s) at either end of an iterative structure is required, regardless of whether they are supported by the language(s) being used.
	An example of definite (count controlled) iteration would be:

	``
	FOR i ← 1 TO 5
	    … Instructions here …
	ENDFOR
	``

	An example of indefinite (condition controlled) iteration with the condition at the start would be:

	``
	WHILE NotSolved
	    … Instructions here …
	ENDWHILE
	``

	Examples of indefinite (condition controlled) iteration with the condition at the end would be:

	``
	REPEAT
	    … Instructions here …
	UNTIL Solved
	``

	``` DO … Instructions here … WHILE NotSolved ```
Use nested selection and nested iteration structures.	An example of nested iteration would be:

	``
	WHILE NotSolved
	    … Instructions here ...
	    FOR i ← 1 TO 5
	        … Instructions here …
	    ENDFOR
	    … Instructions here …
	ENDWHILE
	``

	An example of nested selection would be:

	``
	IF GameWon THEN
	    … Instructions here …
	    IF Score > HighScore THEN
	        … Instructions here …
	    ENDIF
	    … Instructions here …
	ENDIF
	``

Use meaningful identifier names and know why it is important to use them.	Identifier names include names for variables, constants and subroutine names.

3.2.3 Arithmetic operations in a programming language

Content	Additional information
Be familiar with and be able to use: • addition • subtraction • multiplication • real division • integer division, including remainders.	Integer division, including remainders, is usually a two stage process and uses modular arithmetic: eg the calculation 11/2 would generate the following values: Integer division: the integer quotient of 11 divided by 2 (11 DIV 2) = 5 Remainder: the remainder when 11 is divided by 2 (11 MOD 2) = 1

3.2.4 Relational operations in a programming language

Content	Additional information
Be familiar with and be able to use: • equal to • not equal to • less than • greater than • less than or equal to • greater than or equal to.	Students should be able to use these operators within their own programs and be able to interpret them when used within algorithms. Note that different languages may use different symbols to represent these operators.

3.2.5 Boolean operations in a programming language

Content	Additional information
Be familiar with and be able to use: • NOT • AND • OR.	Students should be able to use these operators, and combinations of these operators, within conditions for iterative and selection structures.

3.2.6 Data structures

Content	Additional information
Understand the concept of data structures.	It may be helpful to set the concept of a data structure in various contexts that students may already be familiar with. It may also be helpful to suggest/demonstrate how data structures could be used in a practical setting.

Content	Additional information
Use arrays (or equivalent) in the design of solutions to simple problems.	Only one and two-dimensional arrays are required.
Use records (or equivalent) in the design of solutions to simple problems.	An example of a record definition would be: ```RECORD Car make : String model : String reg : String price : Real noOfDoors : Integer ENDRECORD```

3.2.7 Input/output

Content	Additional information
Be able to obtain user input from the keyboard.	
Be able to output data and information from a program to the computer display.	

3.2.8 String handling operations in a programming language

Content	Additional information
Understand and be able to use: • length • position • substring • concatenation • convert character to character code • convert character code to character • string conversion operations.	Expected string conversion operations: • string to integer • string to real • integer to string • real to string.

3.2.9 Random number generation in a programming language

Content	Additional information
Be able to use random number generation.	Students will be expected to use random number generation within their computer programs. An understanding of how pseudo-random numbers are generated is not required.

3.2.10 Structured programming and subroutines (procedures and functions)

Content	Additional information
Understand the concept of subroutines.	Students should know that a subroutine is a named 'out of line' block of code that may be executed (called) by simply writing its name in a program statement.
Explain the advantages of using subroutines in programs.	
Describe the use of parameters to pass data within programs.	Students should be able to use subroutines that require more than one parameter. Students should be able to describe how data is passed to a subroutine using parameters.
Use subroutines that return values to the calling routine.	Students should be able to describe how data is passed out of a subroutine using return values.
Know that subroutines may declare their own variables, called local variables, and that local variables usually: • only exist while the subroutine is executing • are only accessible within the subroutine.	
Use local variables and explain why it is good practice to do so.	
Describe the structured approach to programming.	Students should be able to describe the structured approach including modularised programming, clear well-documented interfaces (local variables, parameters) and return values. Teachers should be aware that the terms arguments and parameters are sometimes used but in examinable material we will use the term parameter to refer to both of these.
Explain the advantages of the structured approach.	

3.2.11 Robust and secure programming

Content	Additional information
Be able to write simple data validation routines.	Students should be able to use data validation techniques to write simple routines that check the validity of data being entered by a user. The following validation checks are examples of simple data validation routines: • checking if an entered string has a minimum length • checking if a string is empty • checking if data entered lies within a given range (eg between 1 and 10).
Be able to write simple authentication routines.	Students should be able to write a simple authentication routine that uses a username and password. Students will only be required to use plain text usernames and passwords (ie students will not need to encrypt the passwords).
Understand what is meant by testing in the context of algorithms and programs. Be able to correct errors within algorithms and programs.	
Understand what test data is and describe the following types of test data: • normal (typical) • boundary (extreme) • erroneous data.	Boundary data would be for example: If the allowed range is 1 to 10, then boundary data is 0, 1, 10, 11, ie either side of the allowed boundary.
Be able to select and justify the choice of suitable test data for a given problem.	
Understand that there are different types of error: • syntax error • logic error.	
Be able to identify and categorise errors within algorithms and programs.	

3.3 Fundamentals of data representation

3.3.1 Number bases

Content	Additional information
Understand the following number bases: • decimal (base 10) • binary (base 2) • hexadecimal (base 16).	
Understand that computers use binary to represent all data and instructions.	Students should be familiar with the idea that a bit pattern could represent different types of data including text, image, sound and integer.
Explain why hexadecimal is often used in computer science.	

3.3.2 Converting between number bases

Content	Additional information
Understand how binary can be used to represent whole numbers.	Students must be able to represent decimal values between 0 and 255 in binary.
Understand how hexadecimal can be used to represent whole numbers.	Students must be able to represent decimal values between 0 and 255 in hexadecimal.
Be able to convert in both directions between: • binary and decimal • binary and hexadecimal • decimal and hexadecimal.	The following equivalent maximum values will be used: • decimal: 255 • binary: 1111 1111 • hexadecimal: FF

3.3.3 Units of information

Content	Additional information
Know that: • a bit is the fundamental unit of information • a byte is a group of 8 bits.	A bit is either a 0 or a 1. • b represents bit • B represents byte

Content	Additional information
Know that quantities of bytes can be described using prefixes. Know the names, symbols and corresponding values for the decimal prefixes: • kilo, 1 kB is 1,000 bytes • mega, 1 MB is 1,000 kilobytes • giga, 1 GB is 1,000 Megabytes • tera, 1 TB is 1,000 Gigabytes. Be able to compare quantities of bytes using the prefixes above.	Students might benefit from knowing that historically the terms kilobyte, megabyte, etc have often been used to represent powers of 2. The International System of Units (SI units) kilo, mega and so forth refer to values based on powers of 10. When referring to powers of 2 the terms kibi, mebi and so forth would normally be used but students do not need to know these.

3.3.4 Binary arithmetic

Content	Additional information
Be able to add together up to three binary numbers.	Students will need to be able to add together up to three binary numbers using a maximum of 8 bits per number. Students will only be expected to add together a maximum of three 1s in a single column. Answers will be a maximum of 8 bits in length and will not involve carrying beyond the 8th bit.
Be able to apply a binary shift to a binary number.	Students will be expected to use a maximum of 8 bits. Students will be expected to understand and use only a logical binary shift. Students will not need to understand or use fractional representations.
Describe situations where binary shifts can be used.	Binary shifts can be used to perform simple multiplication/division by powers of 2.

3.3.5 Character encoding

Content	Additional information
Understand what a character set is and be able to describe the following character encoding methods: • 7-bit ASCII • Unicode.	Students should be able to use a given character encoding table to: • convert characters to character codes • convert character codes to characters.

Content	Additional information
Understand that character codes are commonly grouped and run in sequence within encoding tables.	Students should know that character codes are grouped and that they run in sequence. For example in ASCII 'A' is coded as 65, 'B' as 66, and so on, meaning that the codes for the other capital letters can be calculated once the code for 'A' is known. This pattern also applies to other groupings such as lower case letters and digits.
Describe the purpose of Unicode and the advantages of Unicode over ASCII. Know that Unicode uses the same codes as ASCII up to 127.	Students should be able to explain the need for data representation of different alphabets and of special symbols allowing a far greater range of characters. It is not necessary to be familiar with UTF-8, UTF-16 or other different versions of Unicode.

3.3.6 Representing images

Content	Additional information
Understand what a pixel is and be able to describe how pixels relate to an image and the way images are displayed.	Students should know that the term pixel is short for picture element. A pixel is a single point in an image.
Describe the following for bitmaps: • image size • colour depth. Know that the size of a bitmap image is measured in pixels (width x height).	The size of an image is expressed directly as width of image in pixels by height of image in pixels using the notation width x height. Colour depth is the number of bits used to represent each pixel.
Describe how a bitmap represents an image using pixels and colour depth.	Students should be able to explain how bitmaps are made from pixels.
Describe using examples how the number of pixels and colour depth can affect the file size of a bitmap image.	Students should be able to describe how higher numbers of pixels and higher colour depths can affect file size, and should also be able to use examples.
Calculate bitmap image file sizes based on the number of pixels and colour depth.	Students only need to use colour depth and number of pixels within their calculations. Size = (bits) = W x H x D Size = (bytes) = (W x H x D)/8 W = image width H = image height D = colour depth in bits.

Content	Additional information
Convert binary data into a bitmap image.	Given a binary pattern that represents a simple bitmap, students should be able to draw the resulting image as a series of pixels.
Convert a bitmap image into binary data.	Given a simple bitmap, students should be able to write down a bit pattern that represents the image.

3.3.7 Representing sound

Content	Additional information
Understand that sound is analogue and that it must be converted to a digital form for storage and processing in a computer.	
Understand that analogue signals are sampled to create the digital version of sound.	Students should understand that a sample is a measure of amplitude at a point in time.
Describe the digital representation of sound in terms of: • sampling rate • sample resolution.	Sampling rate is the number of samples taken in a second and is usually measured in hertz (1 hertz = 1 sample per second). Sample resolution is the number of bits per sample.
Calculate sound file sizes based on the sampling rate and the sample resolution.	File size (bits) = rate x res x secs rate = sampling rate res = sample resolution secs = number of seconds

3.3.8 Data compression

Content	Additional information
Explain what data compression is. Understand why data may be compressed and that there are different ways to compress data.	Students should understand that it is common for data to be compressed and should be able to explain why it may be necessary or desirable to compress data.
Explain how data can be compressed using Huffman coding. Be able to interpret Huffman trees.	Students should be familiar with the process of using a tree to represent the Huffman code.

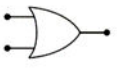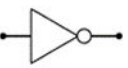| Content	Additional information
Be able to calculate the number of bits required to store a piece of data compressed using Huffman coding. Be able to calculate the number of bits required to store a piece of uncompressed data in ASCII.	Students should be familiar with carrying out calculations to determine the number of bits saved by compressing a piece of data using Huffman coding.
Explain how data can be compressed using run length encoding (RLE).	Students should be familiar with the process of using frequency/data pairs to reduce the amount of data stored.
Represent data in RLE frequency/data pairs.	Students could be given a bitmap representation and they would be expected to show the frequency and value pairs for each row, eg 0000011100000011 would become 5 0 3 1 6 0 2 1.

3.4 Computer systems

3.4.1 Hardware and software

Content	Additional information
Define the terms hardware and software and understand the relationship between them.	

3.4.2 Boolean logic

Content	Additional information
Construct truth tables for the following logic gates: • NOT • AND • OR • XOR.	Students do not need to know about or use NAND and NOR logic gates.
Construct truth tables for simple logic circuits using combinations of NOT, AND, OR and XOR gates. Interpret the results of simple truth tables.	Students should be able to construct truth tables which contain up to three inputs.

Content	Additional information
Create, modify and interpret simple logic circuit diagrams. Students will only need to use NOT, AND, OR and XOR gates within logic circuits. Students will be expected to understand and use the following logic circuit symbols: AND OR NOT XOR	Students should be able to construct simple logic circuit diagrams which contain up to three inputs.

Create and interpret simple Boolean expressions made up of NOT, AND, OR and XOR operations.	Students will be expected to understand and use the following Boolean expression operators:
	
	  . to represent the AND gate
	  + to represent the OR gate
	  $\oplus$ to represent the XOR gate
	  Overbar to represent the NOT gate
	
	For example the expression (A AND B) OR (NOT C) would be represented as:
	
	$(A . B) + \bar{C}$

Create the Boolean expression for a simple logic circuit. Create a logic circuit from a simple Boolean expression.	

3.4.3 Software classification

Content	Additional information
Explain what is meant by: • system software • application software. Give examples of both types of software.	Students should understand that: • system software manages the computer system resources and acts as a platform to run application software • application software is software that performs end-user tasks.

Content	Additional information
Understand the need for, and functions of, operating systems (OS) and utility programs. Understand that the OS handles management of the: • processor(s) • memory • input/output (I/O) devices • applications • security.	

3.4.4 Classification of programming languages and translators

Content	Additional information
Know that there are different levels of programming language: • low-level language • high-level language. Explain the main differences between low-level and high-level languages.	Students should understand that most computer programs are written in high-level languages and be able to explain why this is the case.
Know that machine code and assembly language are considered to be low-level languages and explain the differences between them.	Students should be able to • understand that processors execute machine code and that each type of processor has its own specific machine code instruction set • understand that assembly language is often used to develop software for embedded systems and for controlling specific hardware components • understand that assembly language has a 1:1 correspondence with machine code.
Understand that all programming code written in high-level or assembly languages must be translated. Understand that machine code is expressed in binary 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.	

Content	Additional information
Understand that there are three common types of program translator: • interpreter • compiler • assembler. Explain the main differences between these three types of translator. Understand when it would be appropriate to use each type of translator.	Students will need to know that: • assemblers and compilers translate their input into machine code directly • each line of assembly language is assembled into a single machine code instruction • interpreters do not generate machine code directly (they call appropriate machine code subroutines within their own code to carry out statements).

3.4.5 Systems architecture

Content	Additional information
Explain the role and operation of main memory and the following major components of a central processing unit (CPU) within the Von Neumann architecture: • arithmetic logic unit • control unit • clock • register • bus.	A bus is a collection of wires through which data/signals are transmitted from one component to another. Knowledge of specific registers is not required.
Explain the effect of the following on the performance of the CPU: • clock speed • number of processor cores • cache size.	
Understand and explain the Fetch-Execute cycle.	The CPU continually reads instructions stored in main memory and executes them as required: • fetch: the next instruction is fetched to the CPU from main memory • decode: the instruction is decoded to work out what it is • execute: the instruction is executed (carried out). This may include reading/ writing from/to main memory.

Content	Additional information
Understand the different types of memory within a computer: • RAM • ROM • Cache • Register. Know what the different types of memory are used for and why they are required.	
Understand the differences between main memory and secondary storage. Understand the differences between RAM and ROM.	Students should be able to explain the terms volatile and non-volatile. Main memory will be considered to be any form of memory that is directly accessible by the CPU (except for cache and registers). Secondary storage is considered to be any non-volatile storage mechanism not directly accessible by the CPU.
Understand why secondary storage is required.	
Be aware of different types of secondary storage (solid state, optical and magnetic). Explain the operation of solid state, optical and magnetic storage. Discuss the advantages and disadvantages of solid state, optical and magnetic storage.	Students should be aware that SSDs use electrical circuits to persistently store data but will not need to know the precise details such as use of NAND gates.
Explain the term cloud storage.	Students should understand that cloud storage uses magnetic and/or solid state storage at a remote location.
Explain the advantages and disadvantages of cloud storage when compared to local storage.	
Understand the term embedded system and explain how an embedded system differs from a non-embedded system.	Students must be able to give examples of embedded and non-embedded systems.

3.5 Fundamentals of computer networks

Content	Additional information
Define what a computer network is. Discuss the advantages and disadvantages of computer networks.	
Describe the main types of computer network including: • Personal Area Network (PAN) • Local Area Network (LAN) • Wide Area Network (WAN).	PAN – only Bluetooth needs to be considered. LAN – know that these usually cover relatively small geographical areas. LAN – know that these are often owned and controlled/managed by a single person or organisation. WAN – know that the Internet is the biggest example of a WAN. WAN – know that these usually cover a wide geographic area. WAN – know that these are often under collective or distributed ownership.
Understand that networks can be wired or wireless. Discuss the advantages and disadvantages of wireless networks as opposed to wired networks.	Students should know that wired networks can use different types of cable such as fibre and copper and when each would be appropriate.
Describe the following common LAN topologies: • star • bus.	Students should be able to draw topology diagrams and describe the differences between the two topologies. They should also be able to select the most appropriate topology for a given scenario.
Define the term network protocol.	

Content	Additional information

Explain the purpose and use of common network protocols including:
 	
• Ethernet
• Wi-Fi
• TCP (Transmission Control Protocol)
• UDP (User Datagram Protocol)
• IP (Internet Protocol)
• HTTP (Hypertext Transfer Protocol)
• HTTPS (Hypertext Transfer Protocol Secure)
• FTP (File Transfer Protocol)
• email protocols:
  • SMTP (Simple Mail Transfer Protocol)
  • IMAP (Internet Message Access Protocol).
	Students should know what each protocol is used for (eg HTTPS provides an encrypted version of HTTP for more secure web transactions).
	
	Students should understand that Ethernet is a family of related protocols rather than a single protocol. They do not need to know the individual protocols that make up the Ethernet family.
	
	Students should understand that Wi-Fi is a family of related protocols rather than a single protocol. They do not need to know the individual protocols that make up the Wi-Fi family but they should know that Wi-Fi is a trademark and that the generic term for networks of this nature is WLAN.

Understand the need for, and importance of, network security.	
Explain the following methods of network security: • authentication • encryption • firewall • MAC address filtering.	Students should be able to explain, using examples, what each of these security methods is and when each could be used. Students should understand how these methods can work together to provide a greater level of security. The capabilities of firewalls have changed dramatically in recent years and will continue to do so. Students should be aware that a firewall is a network security device that monitors incoming and outgoing network traffic and decides whether to allow or block specific traffic based on a defined set of security rules. Students should understand that MAC address filtering allows devices to access, or be blocked from accessing a network based on their physical address embedded within the device's network adapter.

Content	Additional information
Describe the 4 layer TCP/IP model: • application layer • transport layer • internet layer • link layer. 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 internet layer.	Students should be able to name the layers and describe their main function(s) in a networking environment. Application layer: this is where the network applications, such as web browsers or email programs, operate. Transport layer: this layer sets up the communication between the two hosts and they agree settings such as the size of packets. Internet layer: addresses and packages data for transmission. Routes the packets across the network. Link layer: this is where the network hardware such as the NIC (network interface card) is located. OS device drivers also sit here. Teachers should be aware that the link layer is sometimes referred to as the network access layer or network interface layer. However, students will not be expected to know these alternative layer names.

3.6 Cyber security

3.6.1 Fundamentals of cyber security

Content	Additional information
Be able to define the term cyber security and be able to describe the main purposes of cyber security.	Students should know that cyber security consists of the processes, practices and technologies designed to protect networks, computers, programs and data from attack, damage or unauthorised access.

3.6.2 Cyber security threats

Content	Additional information
Understand and be able to explain the following cyber security threats: • social engineering techniques • malicious code (malware) • pharming • weak and default passwords • misconfigured access rights • removable media • unpatched and/or outdated software.	Pharming is a cyber attack intended to redirect a website's traffic to a fake website.
Explain what penetration testing is and what it is used for.	Penetration testing is the process of attempting to gain access to resources without knowledge of usernames, passwords and other normal means of access. Students should understand the following two types of penetration testing: • when the person or team testing the system has knowledge of and possibly basic credentials for the target system, simulating an attack from inside the system (a malicious insider) • when the person or team testing the system has no knowledge of any credentials for the target system, simulating an attack from outside the system (an external attack).

3.6.2.1 Social engineering

Content	Additional information
Define the term social engineering. Describe what social engineering is and how it can be protected against. Explain the following forms of social engineering: • blagging (pretexting) • phishing • shouldering (or shoulder surfing).	Students should know that social engineering is the art of manipulating people so they give up confidential information. Blagging is the act of creating and using an invented scenario to engage a targeted victim in a manner that increases the chance the victim will divulge information or perform actions that would be unlikely in ordinary circumstances. Phishing is a technique of fraudulently obtaining private information, often using email or SMS. Shouldering is observing a person's private information over their shoulder eg cashpoint machine PIN numbers.

3.6.2.2 Malicious code (malware)

Content	Additional information
Define the term malware. Describe what malware is and how it can be protected against. Describe the following forms of malware: • computer virus • trojan • spyware.	Malware is an umbrella term used to refer to a variety of forms of hostile or intrusive software.

3.6.3 Methods to detect and prevent cyber security threats

Content	Additional information
Understand and be able to explain the following security measures: • biometric measures (particularly for mobile devices) • password systems • CAPTCHA (or similar) • using email confirmations to confirm a user's identity • automatic software updates.	

3.7 Relational databases and structured query language (SQL)

3.7.1 Relational databases

Content	Additional information
Explain the concept of a database.	
Explain the concept of a relational database.	

Content	Additional information
Understand the following database concepts: • table • record • field • data type • primary key • foreign key. Understand that the use of a relational database facilitates the elimination of data inconsistency and data redundancy.	Note that whilst the terms entity, attribute and entity identifier are more commonly used when an abstract model of a database is being considered, the terms given here will be used for both implementations of and abstract models of databases.

3.7.2 Structured query language (SQL)

Content	Additional information
Be able to use SQL to retrieve data from a relational database, using the commands: • `SELECT` • `FROM` • `WHERE` • `ORDER BY…ASC	DESC`	Exam questions will require that data is extracted from no more than two tables for any one query.

Be able to use SQL to insert data into a relational database using the commands.
 	
```INSERT INTO table_name (column1, column 2 …) 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```	

3.8 Ethical, legal and environmental impacts of digital technology on wider society, including issues of privacy

Content	Additional information
Explain the current ethical, legal and environmental impacts and risks of digital technology on society. Where data privacy issues arise these should be considered.	Exam questions will be taken 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. Students will be expected to understand and explain the general principles behind the issues rather than have detailed knowledge on specific issues. Students should be aware that ordinary citizens normally value their privacy and may not like it when governments or security services have too much access. Students should be aware that governments and security services often argue that they cannot keep their citizens safe from terrorism and other attacks unless they have access to private data.

4 Scheme of assessment

Find past papers and mark schemes, and sample papers for new courses, on our website at aqa.org.uk/pastpapers

This specification is designed to be taken over two years.

This is a linear qualification. In order to achieve the award, students must complete all assessments at the end of the course and in the same series.

GCSE exams and certification for this specification are available for the first time in May/June 2022 and then every May/June for the life of the specification.

All materials are available in English only.

Our GCSE exams in Computer Science include questions that allow students to demonstrate their ability to:

- recall information
- demonstrate programming skills
- apply their knowledge and understanding
- draw together information from different areas of the specification.

4.1 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.

The exams will measure how students have achieved the following 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.

4.2.1 Assessment objective weightings for GCSE Computer Science

Assessment objectives (AOs)	Component weightings (approx %)		Overall weighting (approx %)
	Paper 1	Paper 2	
AO1	4.4	25.6	30
AO2	20	20	40
AO3	25.6	4.4	30
Overall weighting of components	50	50	100

4.3 Assessment weightings

Final marks will be calculated by adding together the scaled marks for each component. Grade boundaries will be set using this total scaled mark. The scaling and total scaled marks are shown in the table below.

Component	Maximum raw mark	Scaling factor	Maximum scaled mark
Paper 1	90	x1	90
Paper 2	90	x1	90
		Total scaled mark:	180

5 Programming skills

A key part of the delivery of this specification is the development of students' programming skills. Throughout their course of study, students must be given the opportunity to design, write, test and refine, using one or more high-level programming language(s) with a textual program definition. In developing these skills schools and colleges are free to choose the context (ie they can be developed in relation to solving a specific problem or to a specification).

In assessments where programming skills are assessed, we will assess students' ability to:

- design
- write
- test, and
- refine

a program to a set task/brief (or to solve a problem). Students are free to use any of the programming languages supported by this specification at the time of their assessment.

5.1 Programming skills authentication

The head of the school or college is responsible for making sure that the programming skills are delivered as an essential part of the course.To meet Ofqual's qualification and subject criteria, schools and colleges must provide a 'Practical programming statement'.

The 'Practical programming statement' is a true and accurate written statement made by each school or college which confirms that it has taken reasonable steps to ensure that each learner has had the opportunity to undertake a programming task or tasks that allows students to develop the required skills.

The 'Practical programming statement' will be provided by us.

5.2 Avoiding malpractice

The school or college must submit to us a 'Practical programming statement' which confirms that all students undertaking this course of study have had the opportunity to develop the skills outlined above.

Failure to complete the 'Practical programming statement' and return it to us in good time will be considered malpractice/maladministration and may result in the school or college being referred to our irregularities team.

6 General administration

You can find information about all aspects of administration, as well as all the forms you need, at aqa.org.uk/examsadmin

6.1 Entries and codes

You only need to make one entry for each qualification – this will cover the question papers and certification.

Every specification is given a national discount (classification) code by the Department for Education (DfE), which indicates its subject area.

If a student takes two specifications with the same discount code:

- further and higher education providers are likely to take the view that they have only achieved one of the two qualifications
- only one of them will be counted for the purpose of the *School and College Performance tables* – the DfE's rules on 'early entry' will determine which one.

Please check this before your students start their course.

Qualification title	Option	AQA entry code	DfE discount code
AQA GCSE in Computer Science	Option A (C#)	8525A	CK1
	Option B (Python)	8525B	CK1
	Option C (VB.NET)	8525C	CK1

This specification complies with:

- Ofqual *General conditions of recognition* that apply to all regulated qualifications
- Ofqual GCSE qualification level conditions that apply to all GCSEs
- Ofqual GCSE subject level conditions that apply to all GCSEs in this subject
- all other relevant regulatory documents.

The Ofqual qualification accreditation number (QAN) is 601/8301/9.

6.2 Overlaps with other qualifications

There are no overlaps with any other AQA qualifications at this level.

6.3 Awarding grades and reporting results

The qualification will be graded on a nine-point scale: 1 to 9 – where 9 is the best grade.

A student taking Foundation tier assessments will be awarded a grade within the range of 1 to 5. Students who fail to reach the minimum standard for grade 1 will be recorded as U (unclassified) and will not receive a qualification certificate.

A student taking Higher tier assessments will be awarded a grade within the range of 4 to 9. A student sitting the Higher tier who just fails to achieve grade 4 will be awarded an allowed grade 3. Students who fail to reach the minimum standard for the allowed grade 3 will be recorded as U (unclassified) and will not receive a qualification certificate.

6.4 Resits and shelf life

Students can resit the qualification as many times as they wish, within the shelf life of the qualification.

6.5 Previous learning and prerequisites

There are no previous learning requirements. Any requirements for entry to a course based on this specification are at the discretion of schools and colleges.

6.6 Access to assessment: diversity and inclusion

General qualifications are designed to prepare students for a wide range of occupations and further study. Therefore our qualifications must assess a wide range of competences.

The subject criteria have been assessed to see if any of the skills or knowledge required present any possible difficulty to any students, whatever their ethnic background, religion, sex, age, disability or sexuality. Tests of specific competences were only included if they were important to the subject.

As members of the Joint Council for Qualifications (JCQ) we participate in the production of the JCQ document *Access Arrangements and Reasonable Adjustments: General and Vocational qualifications*. We follow these guidelines when assessing the needs of individual students who may require an access arrangement or reasonable adjustment. This document is published at jcq.org.uk

Students with disabilities and special needs

We're required by the Equality Act 2010 to make reasonable adjustments to remove or lessen any disadvantage that affects a disabled student.

We can make arrangements for disabled students and students with special needs to help them access the assessments, as long as the competences being tested aren't changed. Access arrangements must be agreed **before** the assessment. For example, a Braille paper would be a reasonable adjustment for a Braille reader.

To arrange access arrangements or reasonable adjustments, you can apply using the online service at aqa.org.uk/eaqa

Special consideration

We can give special consideration to students who have been disadvantaged at the time of the assessment through no fault of their own – for example a temporary illness, injury or serious problem such as family bereavement. We can only do this **after** the assessment.

Your exams officer should apply online for special consideration at aqa.org.uk/eaqa

For more information and advice visit aqa.org.uk/access or email accessarrangementsqueries@aqa.org.uk

6.7 Working with AQA for the first time

If your school or college hasn't previously offered our specifications, you need to register as an AQA centre. Find out how at aqa.org.uk/becomeacentre

6.8 Private candidates

This specification is available to private candidates.

A private candidate is someone who enters for exams through an AQA approved school or college but is not enrolled as a student there.

A private candidate may be self-taught, home schooled or have private tuition, either with a tutor or through a distance learning organisation. They must be based in the UK.

If you have any queries as a private candidate, you can:

- speak to the exams officer at the school or college where you intend to take your exams
- visit our website at aqa.org.uk/privatecandidates
- email privatecandidates@aqa.org.uk

Get help and support

Visit our website for information, guidance, support and resources at

You can talk directly to the Computer Science subject team:

E: computerscience@aqa.org.uk

T: 0161 957 3980

aqa.org.uk