![AQA Realising potential]

GCSE
COMPUTER SCIENCE
(8520)

Specification
For teaching from September 2016 onwards
For exams in 2018 onwards

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 new 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 computational thinking exam to provide an academically challenging programme of study for students of all ability levels. You can choose from a range of programming languages for the programming project, enabling you to tailor your specification to the strengths and preferences of you 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.

Teaching resources

Visit aqa.org.uk/8520 to see all our teaching resources. They include:

- specimen papers and mark schemes to show the standards required and how your students' papers will be marked
- dedicated subject advisors and exemplar specimens to guide you through the programming project
- sample schemes of work and lesson plans to help you plan your course with confidence
- a range of easy-to-use, AQA approved textbooks
- a 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
- subject expertise courses for all teachers, from newly-qualified teachers who are just getting started to experienced teachers looking for fresh inspiration.

Preparing for exams

Visit aqa.org.uk/8520 for everything you need to prepare for our exams, including:

- past papers, mark schemes and examiners' reports
- specimen papers and mark schemes for new courses
- Exampro: a searchable bank of past AQA exam questions
- exemplar 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

Wherever you are in your career, there's always something new to learn. As well as subject-specific training, we offer a range of courses to help boost your skills.

- Improve your teaching skills in areas including differentiation, teaching literacy and meeting Ofsted requirements.
- Prepare for a new role with our leadership and management courses.

You can attend a course at venues around the country, in your school or online – whatever suits your needs and availability. Find out more at coursesandevents.aqa.org.uk

Help and support available

Visit our website for information, guidance, support and resources at aqa.org.uk/8520

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

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 and submit their programming project at the end of the course.

2.1 Subject content

1. Fundamentals of algorithms (page 9)
2. Programming (page 11)
3. Fundamentals of data representation (page 17)
4. Computer systems (page 22)
5. Fundamentals of computer networks (page 25)
6. Fundamentals of cyber security (page 27)
7. Ethical, legal and environmental impacts of digital technology on wider society, including issues of privacy (page 30)
8. Aspects of software development (page 30)
9. Programming project (page 31)

2.2 Assessments

Paper 1: Computational thinking and problem solving
What's assessed
Computational thinking, problem solving, code tracing and applied computing as well as theoretical knowledge of computer science from subject content 1–4 above.
How it's assessed
• Written exam set in practically based scenarios: 1 hour 30 minutes • 80 marks • 50% of GCSE
Questions
A mix of multiple choice, short answer and longer answer questions assessing a student's practical problem solving and computational thinking skills.

✚

Paper 2: Written assessment
What's assessed
Theoretical knowledge from subject content 3–7 above.
How it's assessed
• Written exam: 1 hour 30 minutes • 80 marks • 50% of GCSE
Questions
A mix of multiple choice, short answer, longer answer and extended response questions assessing a student's theoretical knowledge.

+

Programming project
Purpose
The programming project develops a student's ability to use the knowledge and skills gained through the course to solve a problem. Students will be expected to follow a systematic approach to problem solving, consistent with the skills described in Section 8 of the subject content. The skills developed can be applied to exam questions on computational thinking.
What is produced
• A computer program to solve the programming project • Written report: totalling 20 hours of timetabled work
Tasks
The development of a computer program along with the computer programming code itself which has been designed, written and tested by a student to solve a problem. Students will produce an original report outlining this development.

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 3.1–3.7 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.

For the programming project we will support the following programming languages:

- C#, C++, C
- Java
- Pascal/Delphi
- Python (versions 3 and 2)
- VB.Net.

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 and flowcharts.	Any exam question where students are given pseudo-code will use the AQA standard version. However, when students are writing their own pseudo-code they may do so using any form as long as the meaning is clear and unambiguous.

Content	Additional information
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 exams even if the programming language(s) they have been taught do not support all of the topics. Written exams will always present algorithms and code segments using the current version of the AQA pseudo-code document, which can be found on the AQA website, although students can present their answers to questions in any suitable format and do not need to use the AQA pseudo-code when answering questions.

3.2.1 Data types

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

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 imperative programming languages. Students should be able to write programs using these statement types. They should be able to interpret algorithms that include these statement types. Students should know why named constants and variables are used.

Content	Additional information

Use definite and indefinite 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 iteration would be:
	
	``
	FOR i ← 1 TO 5
	    … Instructions here …
	ENDFOR
	``
	
	An example of indefinite iteration with the condition at the start would be:
	
	``
	WHILE NotSolved
	    … Instructions here …
	ENDWHILE
	``
	
	An example of indefinite iteration with the condition at the end would be:
	
	``
	REPEAT
	    … Instructions here …
	UNTIL Solved
	``

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. In assessment material we will use the following symbols: $=, \neq, <, >, \leq, \geq$

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.	

3.2.7 Input/output and file handling

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.	
Be able to read/write from/to a text file.	

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 Subroutines (procedures and functions)

Content	Additional information
Understand the concept of subroutines.	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.	

3.2.11 Structured programming

Content	Additional information
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.	

3.2.12 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).
Be able to select suitable test data that covers normal (typical), boundary (extreme) and erroneous data. Be able to justify the choice of test data.	

3.2.13 Classification of programming languages

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.	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.

Content	Additional information
Understand that ultimately all programming code written in high-level or assembly languages must be translated into machine code. Understand that machine code is expressed in binary and is specific to a processor or family of processors.	
Understand the advantages and disadvantages of low-level language programming compared with high-level language programming.	
Understand that there are three common types of program translator: • interpreter • compiler • assembler. Explain the main differences between these three types of translator. Understand when it would be appropriate to use each type of translator.	

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
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.	Students might benefit from knowing that historically the terms kilobyte, megabyte, etc have often been used to represent powers of 2. The SI units of 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 eight bits.

Content	Additional information
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.
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 a graphical image. VDUs display pictures by dividing the display screen into thousands (or millions) of pixels, arranged into rows and columns.

Content	Additional information
Describe the following for bitmaps: • size in pixels • colour depth. Know that the size of a bitmap image in pixels (width x height) is known as the image resolution.	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.
Convert binary data into a black and white image.	Given a binary pattern that represents a black and white bitmap, students should be able to draw the resulting image as a series of pixels.
Convert a black and white image into binary data.	Given a black and white 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 sound waves are sampled to create the digital version of sound.	Understand that a sample is a measure of amplitude at a point in time.

Content	Additional information
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/create Huffman trees.	Students should be familiar with the process of using a tree to represent the Huffman code. Students should be able to create a new Huffman tree or use a given Huffman tree to: • determine the code used for a particular node within the tree (encoding) • determine the node within a tree given its code (decoding).
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.

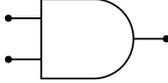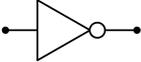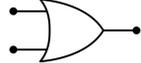| Content	Additional information
Represent data in RLE frequency/data pairs.	Students could be given a bitmap representation and they would be expected to show the frequency and value pairs for each row, eg 0000011100000011 would become 5 0 3 1 6 0 2 1.

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.	Students do not need to know about or use NAND, NOR and XOR logic gates.
Construct truth tables for simple logic circuits. 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 should be able to construct simple logic circuit diagrams which contain up to three inputs. Students will only need to use AND, OR and NOT gates within logic circuits. Students will be expected to understand and use the following logic circuit symbols: AND NOT OR

3.4.3 Software classification

Content	Additional information
Explain what is meant by: • system software • application software. Give examples of both types of software.	
Understand the need for, and functions of, operating systems (OS) and utility programs. Understand that the OS handles management of the: • processor(s) • memory • I/O devices • applications • security.	

3.4.4 Systems architecture

Content	Additional information
Explain the Von Neumann architecture.	
Explain the role and operation of main memory and the following major components of a central processing unit (CPU): • arithmetic logic unit • control unit • clock • bus.	A bus is a collection of wires through which data is transmitted from one component to another. Main memory will be considered to be any form of memory that is directly accessible by the CPU, except for cache and registers.
Explain the effect of the following on the performance of the CPU: • clock speed • number of processor cores • cache size • cache type.	
Understand and explain the Fetch-Execute cycle.	The CPU continuously 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.
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. 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.

Content	Additional information
Explain the term 'cloud storage'.	Students should understand that cloud storage uses magnetic and increasingly 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 benefits and risks 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 benefits and risks of wireless networks as opposed to wired networks.	Know that wired networks can use different types of cable such as fibre and copper and when each would be appropriate.
Explain the following common network topologies: • star • bus.	Students should be able to draw topology diagrams and explain 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. 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 'language' and 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 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.

3.6.1 Cyber security threats

Content	Additional information
Understand and be able to explain the following cyber security threats: • social engineering techniques • malicious code • weak and default passwords • misconfigured access rights • removable media • unpatched and/or outdated software.	
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 that the aim of a white-box penetration test is to simulate a malicious insider who has knowledge of and possibly basic credentials for the target system. Students should understand that the aim of a black-box penetration test is to simulate an external hacking or cyber warfare attack.

3.6.1.1 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 • pharming • 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. Pharming is a cyber attack intended to redirect a website's traffic to another, fake site. Shouldering is observing a person's private information over their shoulder eg cashpoint machine PIN numbers.

3.6.1.2 Malicious code

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 • adware.	Malware is an umbrella term used to refer to a variety of forms of hostile or intrusive software.

3.6.2 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 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 • theft of computer code • issues around copyright of algorithms • cracking • hacking • wearable technologies • computer based implants. 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.

3.8 Aspects of software development

The content in this section will be covered through the programming project.

Content	Additional information
Design Be aware that before constructing a solution, the solution should be designed, for example planning data structures for the data model, designing algorithms, designing an appropriate modular structure for the solution and designing the user interface.	Students should have sufficient experience of successfully structuring programs into modular parts with clear documented interfaces to enable them to design appropriate modular structures for solutions. Students should have sufficient experience of successfully including authentication and data validation systems within their computer programs.
Implementation Be aware that the models and algorithms need to be implemented in the form of data structures and code (instructions) that a computer can understand.	Students should have sufficient practice of 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.
Testing Be aware that the implementation must be tested for the presence of errors, using selected test data covering normal (typical), boundary (extreme) and erroneous data.	Students should have practical experience of 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.
Evaluation/refining Be aware that code created during implementation will often require refining as a result of testing. Be aware of the importance of assessing how well the solution meets the requirements of the problem and how the solution could be improved if the problem were to be revisited.	Students should have practical experience of refining programs in response to testing outcomes. Students should have practical experience of assessing how well their solutions meet the original requirements of the problem. Students should have practical experience of explaining how a solution could be improved if the problem were to be revisited.

3.9 Programming project

3.9.1 Overview

3.9.1.1 Purpose of programming project

The programming project allows students to develop their practical skills in a problem solving context by coding a solution to a given problem and producing a report documenting the development of the solution. The programming project should be treated as a learning experience: allowing students to work independently, over a 20 hour period, extending their programming skills and increasing their understanding of practical, real world applications of computer science.

Additional information can be found in the teachers' notes which accompany the relevant programming project task.

3.9.2 The task

3.9.2.1 Setting the task

We will set the programming project task: this will be available to schools and colleges in September of the final academic year of the course.

The task will change for each new cohort of students.

It is the responsibility of the teacher to make sure that the correct task is used when preparing their students.

3.9.2.2 Taking the task

The task will comprise of a single project which should be undertaken in a period totalling 20 timetabled hours. When completing the task, students must work independently and produce a unique piece of work.

Students must program in one of the high-level programming languages available for use in this specification.

The completed task will generate a:

- program designed, written, tested and refined by the student
- written report.

Each student must produce their own report, in either hard copy or electronic format (saved to CD). The report must show evidence that students have:

- designed their own solution
- created a unique solution
- tested their solution
- indicated potential enhancements and refinements to their solution.

3.9.2.3 Authentication of students' work

Teachers must be confident that the evidence generated by each student is their own work, that students have had the opportunity to complete it in 20 timetabled hours, and that they can authenticate these conditions (see Supervising and authenticating). It is the school or college's responsibility to ensure that the work submitted for monitoring is that of the student.

- Students are **not** allowed to take the programming project home with them.
- Students are **not** allowed to take work on the programming project home to complete.

All work presented for submission **must** have been completed under supervised conditions.

3.9.3 Marking the task

You do not need to mark the work, but you must submit an unmarked sample of your students work for monitoring purposes.

You may choose to mark the task and provide feedback to students to enhance learning, but please do not submit those marks to AQA.

If you do choose to mark the task, you can use the marking criteria in the teacher's notes.

4 Scheme of assessment

Find past papers and mark schemes, and specimen 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 2018 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
- draw together information from different areas of the specification
- apply their knowledge and understanding.

4.1 Aims and learning outcomes

Courses based on this specification should enable 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 mathematical skills relevant to computer science.

4.2 Assessment objectives

Assessment objectives (AOs) are set by Ofqual and are the same across all GCSE Computer Science specifications and all exam boards.

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.

Assessment objective weightings for GCSE Computer Science

Assessment objectives (AOs)	Component weightings (approx %)		Overall weighting (approx %)
	Paper 1	Paper 2	
AO1	7	30	35–40
AO2	28	20	45–50
AO3	15	0	15–20
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	80	x1	80
Paper 2	80	x1	80
		Total scaled mark:	160

5 Programming project administration

This specification includes a programming project.

Visit aqa.org.uk/8520 for detailed information about all aspects of the programming project administration.

The head of the school or college is responsible for making sure that the programming project is conducted in line with our instructions and Joint Council for Qualifications (JCQ) instructions.

To ensure the delivery of the programming project and reduce the likelihood of malpractice, we will monitor students' work.

5.1 Supervising and authenticating

To meet Ofqual's qualification and subject criteria:

- **students** must sign the *Candidate record form* (CRF) to confirm that the work submitted is their own
- all **teachers** who have supervised a student's work must sign the declaration of authentication on the CRF. This is to confirm that the work is that of the student concerned and was conducted under the conditions laid down by this specification
- teachers must ensure that a CRF is provided with each student's work

Schools and colleges must also provide a signed 'Programming project declaration' confirming that each student has:

- had 20 timetabled hours set aside to undertake the programming project
- produced a written account of their programming project which represents their individual work, covers each part of the project and references any resources used and support given.

Students must be subject to supervision to ensure that the work submitted can be confidently authenticated as their own. You are permitted to explain or amplify the language used in the programming project if students are unable to understand what is required, and assist them through the stages of the development program and production of the report to ensure that their work is appropriately focused. However, the program and report should remain the students own work. In providing advice teachers must not provide templates, model answers, writing frames, specific feedback on how to solve the live programming project task or aspects of it.

Please make a note of all the individual support the student received on the CRF. You should also provide a programming project declaration signed by the Head of centre, confirming that the rules for the conduct of the programming project have been adhered to. If either of these are not signed, we cannot accept the student's work.

Teachers are required to keep a log of the number of hours spent on the programming project, this must be provided with the sample of students' work that is sent to AQA. If you fail to provide students with the opportunity of 20 timetabled hours to undertake the programming project, it will be considered malpractice.

5.2 Avoiding malpractice

You must be able to confirm that the work submitted by each student is their own and has been completed within 20 timetabled hours. All work must be completed under supervision in the classroom and appropriate action taken to ensure that students are not able to bring in work produced outside of the supervised time.

Further details on the rules and procedures for the authentication of students work are in the teacher's notes that accompany each programming project task.

If you identify malpractice **before** the student signs the declaration of authentication, you don't need to report it to us. Please deal with it in accordance with your school or college's internal procedures. We expect schools and colleges to treat such cases very seriously.

If you identify malpractice **after** the student has signed the declaration of authentication, the head of your school or college must submit full details of the case to us at the earliest opportunity. Please complete the form *JCQ/M1*, available from the JCQ website at jcq.org.uk

We have agreed a date with Ofqual when the programming project papers may be given to teachers and students. This can be found at aqa.org.uk/timetables

If the programming project task is released before Ofqual's agreed date we will treat this as malpractice.

You must record details of any work which is not the student's own on the CRF or other appropriate place.

You should consult your exams officer about these procedures.

5.3 Submitting work

You are required to submit a sample of students work to AQA on or shortly after 15 May. You must check that the CRF accompanies the work of each student and that the signed programming project declaration is emailed to us at neadeclaration@aqa.org.uk

The deadline for submitting the work of each student is given at aqa.org.uk/keydates

5.4 Factors affecting individual students

For advice and guidance about arrangements for any of your students, please email us as early as possible at eos@aqa.org.uk

Occasional absence: you should be able to accept the occasional absence of students by making sure they have the chance to make up what they have missed. You may organise an alternative supervised session for students who were absent at the time you originally arranged.

Lost work: if work is lost you must tell us how and when it was lost and who was responsible, using our special consideration online service at aqa.org.uk/eaqa

Special help: where students need special help which goes beyond normal learning support,pleaseuse the CRF to tell us so that this help can be taken into account during monitoring

Students who move schools: students who move from one school or college to another during the course sometimes need additional help to meet the requirements. How you deal with this depends on when the move takes place. If it happens early in the course, the new school or

college should be responsible for the work. If it happens late in the course, it may be possible to arrange for AQA to monitor the work as a student who was 'Educated Elsewhere'.

5.5 Keeping students work

Students' work must be kept under secure conditions from the time that it is submitted, with completed CRF. After the monitoring period and the deadline for Enquiries about Results (or once any enquiryis resolved) you may return the work to students.

5.6 Monitoring work

We'll contact you to let you know which students' work to submit. If you are entering fewer than 20 students (or submitting work electronically) it will be the work of all your students. Otherwise it will be a percentage of your students' work.

An AQA monitor will check a sample of your students work to ensure that:

- students have had the opportunity to spend 20 timetabled hours on their programming project
- students have had the opportunity to complete all sections of the programming project and have produced both a program and written report
- the work is that of the individual student.

Work will not be returned to your school or college after monitoring has taken place.

School and college consortia

If you are in a consortium of schools or colleges with joint teaching arrangements (where students from different schools and colleges have been taught together but entered through the school or college at which they are on roll), you must let us know by:

- filling in the *Application for Centre Consortium Arrangements for centre-assessed work*, which is available from the JCQ website jcq.org.uk
- appointing a consortium co-ordinator who can speak to us on behalf of all schools and colleges in the consortium. If there are different co-ordinators for different specifications, a copy of the form must be sent in for each specification.

We will allocate the same monitor to all schools and colleges in the consortium and treat the students as a single group for monitoring purposes.

All the work must be available at the lead school or college.

5.7 After monitoring

We will not return your students' work. You will not receive a report on the outcome of monitoring when the results are issued.

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 all the question papers, programming project task 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	AQA entry code	DfE discount code
AQA GCSE in Computer Science	8520	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.

Students who fail to reach the minimum standard for grade 1 will be recorded as U (unclassified) and will not receive a qualification certificate.

6.4 Re-sits and shelf life

Students can re-sit 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. If any difficulties were encountered, the criteria were reviewed again to make sure that 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 on the JCQ website at jcq.org.uk

6.6.1 Students with disabilities and special needs

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 are not changed. Access arrangements must be agreed **before** the assessment. For example, a Braille paper would be a reasonable adjustment for a Braille reader but not for a student who does not read Braille.

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

If you have students who need access arrangements or reasonable adjustments, you can apply using the Access arrangements online service at aqa.org.uk/eaqa

6.6.2 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 the death of a relative. 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 about access arrangements, reasonable adjustments and special consideration please see aqa.org.uk/access or email accessarrangementsqueries@aqa.org.uk

6.7 Working with AQA for the first time

If your school or college has not previously offered any AQA specification, you need to register as an AQA centre to offer our specifications to your students. Find out how at aqa.org.uk/becomeacentre

6.8 Private candidates

This specification is not available to private candidates

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