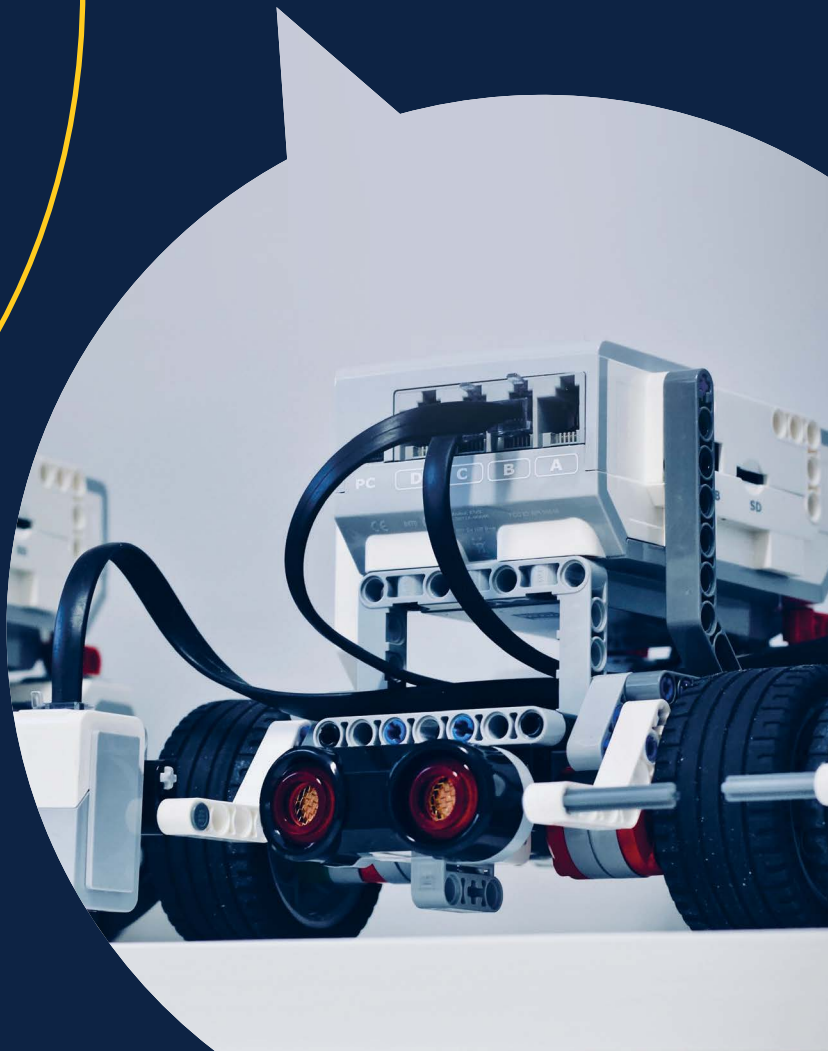
OXFORD AQA
INTERNATIONAL QUALIFICATIONS

International GCSE

Computer Science

(9210) Specification



For teaching from September 2017 onwards

For exams May/June 2019 onwards

For teaching and examination outside the United Kingdom

Contents

1 Introduction **5**

1.1 Why choose OxfordAQA International GCSEs? 5

1.2 Why choose our International GCSE Computer Science? 5

1.3 Recognition 5

1.4 The Oxford International Programme learner attributes 6

1.5 Support and resources to help you teach 7

2 Specification at a glance **8**

2.1 Subject content 8

2.2 Assessments 9

3 Subject content **10**

3.1 Algorithms 10

3.2 Programming 12

3.3 Data representation 19

3.4 Computer systems 24

3.5 Computer networks 27

3.6 Cyber security 29

3.7 Relational databases and structured query language (SQL) 31

3.8 Web page design 32

4 Scheme of assessment **34**

4.1 Aims and learning outcomes 34

4.2 Assessment Objectives 35

4.3 Assessment weightings 35

5 General administration **36**

5.1 Entries and codes 36

5.2 Overlaps with other qualifications 36

5.3 Awarding grades and reporting results 36

5.4 Resits 36

5.5 Previous learning and prerequisites 36
5.6 Access to assessment: equality and inclusion 37
5.7 Working with OxfordAQA for the first time? 37
5.8 Private candidates 37

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 **oxfordaqa.com/9210**
- We will write to you if there are significant changes to the specification.

1 Introduction

1.1 Why choose OxfordAQA International GCSEs?

Our international qualifications enable schools that follow an English curriculum to benefit from the best education expertise in England.

Our International GCSEs offer the same rigour and high quality as GCSEs in England and are relevant and appealing to students worldwide. They reflect a deep understanding of the needs of teachers and schools around the globe and are brought to you by Oxford University Press and AQA, the UK's leading awarding body.

Providing fair, valid and reliable assessments, these qualifications are based on over 100 years of experience, academic research and international best practice. They have been independently validated as being of the same standard as the qualifications accredited by England's examinations regulator, Ofqual. They reflect the latest changes to the English system, enabling students to progress to higher education with up-to-date qualifications.

You can find out about OxfordAQA at **oxfordaqa.com**

1.2 Why choose our International GCSE Computer Science?

The OxfordAQA International GCSE takes a highly practical approach to teaching computer science. Students will spend the majority of the course learning to design, write and test computer programs using a high-level programming language. They will also have the opportunity to cover other useful practical skills such as web page design using HTML and working with relational databases using SQL.

The course content has been selected to provide students with the key skills required to go on to follow further qualification or in the workplace, to allow students to be creative and to give them an understanding of important aspects of computer science beyond programming.

Programming skills are assessed by a practical programming exam with a pre-release skeleton program, which students can complete in the programming language that they have used throughout the course. AQA has considerable experience of successfully delivering computer science qualifications, including written on-screen programming exams in England.

You can find out about all our International GCSE Computer Science qualifications at **oxfordaqa.com/computerscience**

1.3 Recognition

OxfordAQA meet the needs of international students. They are an international alternative and comparable in standard to the Ofqual regulated qualifications offered in the UK. Our qualifications have been independently benchmarked by UK ENIC, the UK national agency for providing expert opinion on qualifications worldwide. They have confirmed they can be considered 'comparable to the overall GCE A-level and GCSE standard offered in the UK'.

To read their report and see the latest list of universities who have stated they accept these international qualifications, visit **oxfordaqa.com/recognition**

1.4 The Oxford International Programme learner attributes

In order to equip students with the skills they need for success both now and in the future, we have worked with Oxford University Press to create the Oxford International Programme. This combines the Oxford International Curriculum with OxfordAQA qualifications, creating an integrated offer for international schools, from Early Years to A-level.

At its core we have introduced the Oxford International Programme learner attributes – the skills and competencies that enable our students to thrive academically, socially and personally.

The learner attributes, alongside our focus on demonstrating higher order critical thinking skills, ensure that students are equipped to get the grades that will take them places, and build the skills they need to be successful when they get there.

1.5 Support and resources to help you teach

We know that support and resources are vital for your teaching and that you have limited time to find or develop good quality materials. That's why we've worked with experienced teachers to provide resources that will help you confidently plan, teach and prepare for exams.

Teaching resources

You will have access to:

- sample scheme of work with resource list to help you plan your course with confidence
- training courses to help you deliver our qualifications
- student textbooks that have been checked and approved by us
- engaging worksheets and activities developed by teachers, for teachers
- switching guide
- teacher guide that includes pseudocode guide, programming language comparison and command words
- teachers' notes.

Preparing for exams

You will have access to the support you need to prepare for our exams, including:

- specimen papers and mark schemes
- exemplar student answers with examiner commentaries.

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

After the first examination series, you can use this tool to see which questions proved to be 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.

Information about results, including maintaining standards over time, grade boundaries and our post-results services, will be available on our website in preparation for the first examination series.

Help and support

Visit our website for information, guidance, support and resources at **oxfordaqa.com/9210**

You can contact the subject team directly at **info@oxfordaqa.com** or call us on +44 (0)161 696 5995 (option 1 and then 1 again).

Please note: We aim to respond to all email enquiries within two working days.

Our UK office hours are Monday to Friday, 8am – 5pm.

2 Specification at a glance

The title of the qualification is:

- OxfordAQA International GCSE Computer Science.

2.1 Subject content

1 Algorithms (page 10)

2 Programming (page 12)

3 Data representation (page 19)

4 Computer systems (page 24)

5 Computer networks (page 27)

6 Cyber security (page 29)

7 Relational databases and structured query language (SQL) (page 31)

8 Web page design (page 32).

2.2 Assessments

Paper 1: Programming

What's assessed

Writing and testing computer programs, understanding programming concepts and being able to analyse problems in computational terms.

How it's assessed

On-screen programming exam based on a pre-released skeleton program.

Length of exam: 2 hours.

80 marks.

50% weighting.

The pre-released skeleton program will be made available to centres on the 1st of March (May/June examination) and 1st August (November examination) of the year of the examination.

The assessment will be available in the following languages:

- C#
- Python 3
- Visual Basic.

Questions

The question paper will have the following structure:

Section A: Questions about programming that test programming concepts and some non-programming aspects of the skeleton program.

Section B: Short programming questions which require students to make small modifications to the skeleton program, e.g. correcting errors or writing up to several lines of program code.

Section C: Longer programming questions which require students to make major modifications to the skeleton program and will typically require more analysis than Section B questions, eg writing new subroutines, adding significant new functionality to the skeleton program.

Responses to questions will be recorded in a word-processed document, known as the **electronic answer document.**

+

Paper 2: Concepts and principles of computer science

What's assessed

Knowledge and understanding of the key concepts and principles of computer science.

How it's assessed

Written exam.

Length of exam: 2 hours.

80 marks.

50% weighting.

Questions

A mixture of question types including multiple choice, short and longer answer questions.

3 Subject content

3.1 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 pseudocode and flowcharts.	Any exam question where students are given pseudocode will use the Oxford International AQA standard version. However, when students are writing their own pseudocode they may do so using any form as long as the meaning is clear and unambiguous.
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 and be able to follow and write pseudo-code for it.
Understand and explain how the binary search algorithm works.	Students should know the mechanics of the iterative version of the algorithm and be able to follow and write pseudocode for it.
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 recursive version of the algorithm. It will be sufficient for students to explain this algorithm in prose; they will not be expected to be able to write pseudocode for it. Students should be able to demonstrate how a merge sort would be performed on a given set of data.
Understand and explain how the bubble sort algorithm works.	Students should know the mechanics of the algorithm and be able to follow and write pseudo-code for it. Students will be expected to know the version of the algorithm that uses two nested loops, with the outer loop being indefinitely controlled by a condition that tests if any swaps were made and the inner loop being controlled definitely.
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 OxfordAQA' pseudocode, which can be found in the teaching guidance on the OxfordAQA website, although students can present their answers to questions in any suitable format and do not need to use the pseudocode 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 data 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 or not 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. In assessment material we will use the following symbols: =, ≠, <, >, ≤, ≥	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.
Use arrays (or equivalent) in the design of solutions to simple problems.	Only one and two dimensional arrays are required. In Python, a list is a suitable alternative to an array for this specification.

Use records (or equivalent) in the design of solutions to simple problems.	In C#, structs can be used to create records.
	
	In Python, classes can be used in a non-object-oriented way to create records. For example:
	
	`class Coordinate():`
	`  def __init__(self):`
	`  self.x = 0`
	`  self.y = 0`
	
	`myposition = Coordinate()`
	
	`myposition.x = 10`
	`myposition.y = 5`
	
	In Visual Basic, structures can be used to create records.

3.2.7 Input/output and file handling

Content	Additional information
Be able to obtain user input from a 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.	
Use and 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.

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 numeric 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 be required to use only 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 and erroneous data. Be able to justify the choice of test data.	

3.2.13 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 language and explain why this is the case. 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).
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.
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 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 computers use binary to represent data and instructions	
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 may benefit from knowing that historically the terms kilobyte, megabyte, etc have often been used to represent powers of 2. In assessments we will use powers of 10 and the references to powers of 2 will not be used.

3.3.4 Binary arithmetic

Content	Additional information
Be able to add together two binary numbers.	Students will be expected to use a maximum of 8 bits. Answers will be a maximum of 8 bits in length and will not involve carrying beyond the eighth 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.
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.
Describe the following for bitmaps: • size in pixels • colour depth.	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 an image.	Given a binary pattern that represents a bitmap, students should be able to draw the resulting image as a series of pixels.
Convert an image into binary data.	Given a 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.
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 (Hz) x res x secs rate = sampling rate (hz) res = sample resolution secs = number of seconds.

3.3.8 Data compression

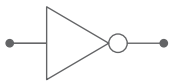
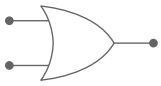
Content	Additional information
Explain what data compression is. Understand why data may be compressed and that there are different ways to compress data.	Students should understand that it is common for data to be compressed and should be able to explain why it may be necessary or desirable to compress data.
Explain how data can be compressed using Huffman coding. Be able to build a Huffman tree. Be able to interpret a Huffman tree.	Students should be familiar with the process of using a tree to represent the Huffman code. Students should be able to build a Huffman tree for a given string of data and show how the string would be compressed using the tree. Students should be able to interpret a given Huffman tree to determine the code used for a particular node within the tree.
Calculate the number of bits required to store a piece of data compressed using Huffman coding. 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.  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 and that use only combinations of NOT, AND and OR gates.
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 NOT, AND and OR gates within logic circuits. Students will be expected to understand and use the logic circuit symbols shown above.
Be able to write a Boolean expression to represent a logic circuit and to draw a logic circuit that implements a Boolean expression.	The following notation will be used for Boolean expressions: • NOT: $\overline{A}$ • AND: $A \cdot B$ • OR: $A + B$

3.4.3 Software classification

Content	Additional information
Explain what is meant by: • system software • application software. Give examples of both types of software.	System software manages the computer system resources. It also acts as a platform to run application software. Application software is software that performs end-user tasks.
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
• the following major components of a central processing unit (CPU):
  ◦ arithmetic logic unit
  ◦ control unit
  ◦ clock
• buses.
	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).
	Secondary storage is considered to be any non-volatile storage mechanism which is not directly accessible by the CPU.

Explain the effect of the following on the performance of the CPU: • clock speed • number of processor cores • cache size.	

Content	Additional information
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.
Understand the differences between main memory and secondary storage. Understand the differences between RAM and ROM. Be able to explain the terms volatile and non-volatile.	Secondary storage is considered to be any non-volatile storage mechanism which is 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 solid state devices (SSDs) use electrical circuits to persistently store data but will not need to know the precise details such as use of NAND gates. Students should be able to compare the advantages and disadvantages of the secondary storage types in comparison to each other and should be able to assess their suitability for a given use/scenario.
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 Computer networks

Content	Additional information
Define what a computer network is. Discuss the benefits and risks of computer networks.	
Describe the LAN and WAN types of computer network: • Local Area Network (LAN) • Wide Area Network (WAN).	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 physical network topologies: • star • bus.	Students should be able to draw topology diagrams and explain the differences between the two topologies.
Define the term 'protocol'.	A protocol is a set of rules that govern how data is transmitted and received by devices.

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.	

Content	Additional information
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.
Describe the 4 layer TCP/IP model: • application layer • transport layer • network 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 network 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. Network 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 network layer is sometimes referred to as the internet layer and that the link layer is sometimes referred to as the network interface layer. However, students will not be expected to know these alternative layer names.

3.6 Cyber security

Content	Additional information
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 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).

3.6.1.1 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 cyberattack 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 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.	
Understand the following important 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 Database design

Content	Additional information
Be able to choose appropriate data types and (where appropriate) lengths for fields in a table.	
Be able to select a suitable primary key for a table.	
Be able to produce a design for a relational database from a description of a scenario.	Scenarios used in examinations will require the use of no more than three tables. Table designs will be written in the form TableName (Field1, Field2, Field3, ….) with underlining used to indicate the primary key.

3.7.3 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.	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 INSERT INTO. Be able to use the SQL commands UPDATE and DELETE FROM to edit and delete data in a database.	Only the form of the INSERT INTO command that specifies all of the values in a record is required.

3.8 Web page design

3.8.1 Key concepts

Content	Additional information
Know that web pages can be constructed using HTML and CSS.	
Know that HTML is used to create the structure of a web page and that CSS can be used to change the style of the page. Understand that server-side scripting languages can be used on a web server for dynamic generation of web pages. Understand that client-side scripting languages can be used to add additional functionality to web pages.	

3.8.2 Hypertext Markup Language (HTML)

Content	Additional information
Be able to create the structure of a web page using HTML.	Students will be expected to be able to use the following HTML tags: General: <html> </html>, <title> </title>, <body> </body>, <style> </style> Block-level: , , <h#> </h#>, , , Inline: , , , , <a> , <img />
Be able to use basic CSS to control the layout of a web page.	A style rule is made up of three parts, a selector, a property and a value: selector {property : value } Two kinds of selector need to be known: type, class. The following properties should be known: background-color, color, text-align, font-family, font-size, font-weight, font-style. Exam questions will only use embedded style sheets, but students may wish to use external style sheets when completing practical work.
Be aware of the key differences between HTML5 and earlier versions of HTML.	Students should know that HTML5: • Supports many facilities that were previously only available by using add-ons to HTML. • Improved provision for the development of web applications through support for scripting and access to many new APIs. • Ability to more easily incorporate graphics and multimedia elements, for example using the <svg>, <canvas>, <audio> and <video> tags. • Use of new semantic elements such as <header> and <article> to define different parts of a web page.

4 Scheme of assessment

You can find mark schemes, and specimen papers for new courses, on our website at **oxfordaqa.com/9210**

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.

Our International GCSE exams and certification for this specification are available for the first time in May/June 2019 and then every May/June and November for the life of the specification.

All materials are available in English only.

Our International GCSE Computer Science includes questions that allow students to demonstrate their ability to:

- recall information
- apply their knowledge and understanding
- write, test and explain program code.

4.1 Aims and learning outcomes

Our International GCSE Computer Science encourages students to be inspired, motivated and challenged by following a broad, coherent, practical, satisfying and worthwhile course of study. It should encourage students to develop their curiosity about the living world, enable students to engage with computer science in their everyday lives in order to make informed choices about further study in computer science and related disciplines.

Our International GCSE Computer Science should enable students to:

- analyse problems in computational terms
- design and write computer programs to solve problems
- test and debug programs
- build simple web pages and work with relational databases
- demonstrate and apply knowledge and understanding of the key concepts and principles of computer science
- progress to employment or further courses of study.

4.2 Assessment Objectives

The assessments 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 in order to develop and test programmed solutions and demonstrate an understanding of programming concepts.

4.2.1 Assessment Objective weightings

Assessment Objectives (AOs)	Component weightings (approx %)		Overall weighting of AOs (approx %)
	Paper 1	Paper 2	
AO1	0	20–25	20–25
AO2	0–5	20–25	20–30
AO3	45–50	0–5	45–55
Overall weighting of units (%)	50	50	100

4.3 Assessment weightings

Component	Maximum raw mark	Scaling factor	Maximum scaled mark
Paper 1	80	1	80
Paper 2	80	1	80
		Total scaled mark:	160

5 General administration

We are committed to delivering assessments of the highest quality and have developed practices and procedures that support this aim. To ensure that all students have a fair experience, we have worked with other awarding bodies in England to develop best practice for maintaining the integrity of assessments. This is published through the Joint Council for Qualifications (JCQ). We will maintain the same high standard through their use for OxfordAQA.

More information on all aspects of administration is available at **oxfordaqa.com/exams-administration**

For any immediate enquiries please contact **info@oxfordaqa.com**

Please note: We aim to respond to all email enquiries within two working days.

Our UK office hours are Monday to Friday, 8am – 5pm local time.

5.1 Entries and codes

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

Qualification title	OxfordAQA entry code
OxfordAQA International GCSE Computer Science	9210

Please check the current version of the Entry Codes book and the latest information about making entries on **oxfordaqa.com/exams-administration**

Exams will be available May/June and in November.

5.2 Overlaps with other qualifications

This specification overlaps with the AQA UK GCSE in Computer Science (8520).

5.3 Awarding grades and reporting results

In line with English GCSEs, this 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.

To find out more about the new grading system, visit our website at **oxfordaqa.com**

5.4 Resits

Candidates can retake the whole qualification as many times as they wish. This is a traditional linear specification; individual components cannot be resat.

5.5 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.

5.6 Access to assessment: equality and inclusion

Our general qualifications are designed to prepare students for a wide range of occupations and further study whilst assessing a wide range of competences.

The subject criteria have been assessed to ensure they test specific competences. The skills or knowledge required do not disadvantage particular groups of students.

Exam access arrangements are available for students with disabilities and special educational needs.

We comply with the *UK Equality Act 2010* to make reasonable adjustments to remove or lessen any disadvantage that affects a disabled student. Information about access arrangements will be issued to schools when they become OxfordAQA centres.

5.7 Working with OxfordAQA for the first time?

You will need to apply to become an OxfordAQA centre to offer our specifications to your students. Find out how at **oxfordaqa.com/centreapprovals**

5.8 Private candidates

Centres may accept private candidates for examined units/components only with the prior agreement of OxfordAQA. If you are an approved OxfordAQA centre and wish to accept private candidates, please contact OxfordAQA at: **info@oxfordaqa.com**

Centres accepting private candidates must provide access to the paper 1 pre-release material when it is released

Private candidates may also enter for examined only units/components via the British Council; please contact your local British Council office for details.

OXFORD AQA
INTERNATIONAL QUALIFICATIONS

Fairness *first*

Thank you for choosing OxfordAQA, the international exam board that puts fairness first.

Benchmarked to UK standards, our exams only ever test subject ability, not language skills or cultural knowledge.

This gives every student the best possible chance to show what they can do and get the results they deserve.

Get in touch

You can contact us at *oxfordaqa.com/contact-us*

or email *info@oxfordaqa.com*

Copyright © 2025 Oxford International AQA Examinations and its licensors. All rights reserved.

Oxford International AQA Examinations retains the copyright on all its publications, including specifications. However, approved OxfordAQA schools are permitted to copy material from this specification for their own internal use.