

GCSE (9-1)

Specification

COMPUTER SCIENCE

J277

For first assessment in 2022

Disclaimer

Specifications are updated over time. Whilst every effort is made to check all documents, there may be contradictions between published resources and the specification, therefore please use the information on the latest specification at all times. Where changes are made to specifications these will be indicated within the document, there will be a new version number indicated, and a summary of the changes. If you do notice a discrepancy between the specification and a resource please contact us at: resources.feedback@ocr.org.uk

We will inform centres about changes to specifications. We will also publish changes on our website. The latest version of our specifications will always be those on our website (ocr.org.uk) and these may differ from printed versions.

© 2024 OCR. All rights reserved.

Copyright

OCR retains the copyright on all its publications, including the specifications. However, registered centres for OCR are permitted to copy material from this specification booklet for their own internal use.

Oxford Cambridge and RSA is a Company Limited by Guarantee. Registered in England. Registered company number 3484466.

Registered office:
The Triangle Building
Shaftesbury Road
Cambridge
CB2 8EA

OCR is an exempt charity.

Contents

1	Why choose an OCR GCSE (9–1) in Computer Science?	2
1a.	Why choose an OCR qualification?	2
1b.	Aims and learning outcomes	3
1c.	What are the key features of this specification?	3
1d.	How do I find out more information?	3
2	The specification overview	5
2a.	OCR’s GCSE (9–1) in Computer Science (J277)	5
2b.	Content of Computer systems (J277/01)	6
2c.	Content of Computational thinking, algorithms and programming (J277/02)	15
2d.	Practical Programming skills	22
2e.	Prior knowledge, learning and progression	22
3	Assessment of GCSE (9–1) in Computer Science	23
3a.	Forms of assessment	23
3b.	Assessment of Practical Programming skills: Component 2	24
3c.	OCR Exam Reference Language	25
3d.	Command words	32
3e.	Assessment Objectives	34
3f.	Total qualification time	34
3g.	Qualification availability outside of England	34
3h.	Language	35
3i.	Assessment availability	35
3j.	Retaking the qualification	35
3k.	Assessment of extended response	35
3l.	Mathematical skills requirement	35
3m.	Synoptic assessment	35
3n.	Calculating qualification result	35
4	Admin: what you need to know	36
4a.	Pre-assessment	36
4b.	Special consideration	37
4c.	External assessment arrangements	37
4d.	Practical Programming skills administration requirements	38
4e.	Results and certificates	38
4f.	Post-results services	39
4g.	Malpractice	39
5	Appendices	40
5a.	Grade descriptors	40
5b.	Overlap with other qualifications	41
5c.	Accessibility	41
5d.	Mathematical skills requirement	41
6	Summary of updates	42
	J277: Summary of updates	42
	J276 to J277: Summary of updates – key changes	44
7	Pathways for Computing	45

1 Why choose an OCR GCSE (9–1) in Computer Science?

1a. Why choose an OCR qualification?

1

Choose OCR and you've got the reassurance that you're working with one of the UK's leading exam boards. Our GCSE (9–1) in Computer Science has been developed in consultation with teachers, employers and Higher Education to provide students with a qualification that's relevant to them and meets their needs.

We're part of the Cambridge Assessment Group, Europe's largest assessment agency and a department of the University of Cambridge. Cambridge Assessment plays a leading role in developing and delivering assessments throughout the world, operating in over 150 countries.

We work with a range of education providers, including schools, colleges, workplaces and other institutions in both the public and private sectors. Over 13,000 centres choose our A Levels, GCSEs and vocational qualifications including Cambridge Nationals and Cambridge Technicals.

Our Specifications

We believe in developing specifications that help you bring the subject to life and inspire your students to achieve more.

We've created teacher-friendly specifications based on extensive research and engagement with the teaching community. They're designed to be straightforward and accessible so that you can tailor the delivery of the course to suit your needs.

Our Support

We provide a range of support services designed to help you at every stage, from preparation through to the delivery of our specifications. This includes:

- a wide range of high-quality creative resources including access to resources provided by leading organisations within the industry

- textbooks available from a number of leading publishers we have worked with. For more information on our publishing partners and their resources visit ocr.org.uk/qualifications/resource-finder/publishing-partners
- Professional Development for teachers to fulfil a range of needs. To join our training (either face-to-face or online) or to search for training materials, you can find what you're looking for at www.ocr.org.uk/qualifications/professional-development
- Active Results – our free results analysis service to help you review the performance of individual students or whole schools
- [ExamBuilder](#) – our online past papers service that enables you to build your own test papers from past OCR exam questions.

Subject Advisors

OCR Subject Advisors provide specialist advice, guidance and support to centres related to our specification, as well as updates on resources and professional development opportunities. Our Subject Advisors work with subject communities through a range of networks to ensure the sharing of ideas and expertise to support teachers.

Keep up to date with OCR

To receive the latest information about any of our qualifications, please register for email updates at: ocr.org.uk/updates

All GCSE (9–1) qualifications offered by OCR are accredited by Ofqual, the Regulator for qualifications offered in England.

The accreditation number for OCR's GCSE (9–1) in Computer Science is QN 601/8355/X.

1b. Aims and learning outcomes

OCR's GCSE (9–1) in Computer Science will encourage students to:

- 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.

1c. What are the key features of this specification?

The key features of OCR's GCSE (9–1) in Computer Science for you and your students are:

- a simple and intuitive assessment model, consisting of two papers, one focusing on computer systems and one with a focus on programming, computational thinking, and algorithms. Both papers have identical weighting and mark allocations
- a specification developed with teachers specifically for teachers. The specification lays out the subject content clearly
- a flexible support package formed after listening to teachers' needs. The support package will enable teachers to easily understand the requirements of the qualification and how it is assessed
- a team of OCR Subject Advisors who support teachers directly and manage the qualification nationally
- the specification has been designed to transition seamlessly into Computer Science at AS Level and/or A Level.

This specification/qualification will enable students to develop:

- valuable thinking and programming skills that are extremely attractive in the modern workplace
- a deep understanding of computational thinking and how to apply it through a chosen programming language.

1d. How do I find out more information?

1

Whether you are an existing OCR centre, or new to OCR and would like to start delivering this course, please visit www.ocr.org.uk. Or you can contact us directly by email or phone.

Contact details:

Email: computerscience@ocr.org.uk

Subject web page: www.ocr.org.uk/computing

Twitter: [@ocr_ict](https://twitter.com/ocr_ict)

Customer Contact Centre: 01223 553998

2 The specification overview

2a. OCR's GCSE (9–1) in Computer Science (J277)

Students take J277/01 and J277/02 to be awarded the OCR GCSE (9–1) in Computer Science.

Content Overview

J277/01: Computer systems

This component will assess:

- 1.1 Systems architecture
- 1.2 Memory and storage
- 1.3 Computer networks, connections and protocols
- 1.4 Network security
- 1.5 Systems software
- 1.6 Ethical, legal, cultural and environmental impacts of digital technology

J277/02: Computational thinking, algorithms and programming

This component will assess:

- 2.1 Algorithms
- 2.2 Programming fundamentals
- 2.3 Producing robust programs
- 2.4 Boolean logic
- 2.5 Programming languages and Integrated Development Environments

Assessment Overview

Written paper: 1 hour and 30 minutes

50% of total GCSE

80 marks

This is a non-calculator paper.

All questions are mandatory.

This paper consists of multiple choice questions, short response questions and extended response questions.

Written paper: 1 hour and 30 minutes

50% of total GCSE

80 marks

This is a non-calculator paper.

This paper has two sections: Section A and Section B. Students must answer both sections.

All questions are mandatory.

In Section B, questions assessing students' ability to write or refine algorithms must be answered using **either** the OCR Exam Reference Language **or** the high-level programming language they are familiar with.

2



Practical Programming

All students must be given the opportunity to undertake a programming task(s), either to a specification or to solve a problem (or problems), during their course of study. Students may draw on some of the content in both components when engaged in Practical Programming.

Please see Sections 2d and 4d for further information.

2b. Content of Computer systems (J277/01)

1.1 – Systems architecture	
Sub topic	Guidance
1.1.1 Architecture of the CPU	
<input type="checkbox"/> The purpose of the CPU: <ul style="list-style-type: none"> ○ The fetch-execute cycle <input type="checkbox"/> Common CPU components and their function: <ul style="list-style-type: none"> ○ ALU (Arithmetic Logic Unit) ○ CU (Control Unit) ○ Cache ○ Registers <input type="checkbox"/> Von Neumann architecture: <ul style="list-style-type: none"> ○ MAR (Memory Address Register) ○ MDR (Memory Data Register) ○ Program Counter ○ Accumulator 	<p>Required</p> <ul style="list-style-type: none"> ✓ What actions occur at each stage of the fetch-execute cycle ✓ The role/purpose of each component and what it manages, stores, or controls during the fetch-execute cycle ✓ The purpose of each register, what it stores (data or address) ✓ The difference between storing data and an address <p>Not required</p> <ul style="list-style-type: none"> ✗ Knowledge of passing of data between registers in each stage
1.1.2 CPU performance	
<input type="checkbox"/> How common characteristics of CPUs affect their performance: <ul style="list-style-type: none"> ○ Clock speed ○ Cache size ○ Number of cores 	<p>Required</p> <ul style="list-style-type: none"> ✓ Understanding of each characteristic listed ✓ The effects of changing any of the common characteristics on system performance, either individually or in combination
1.1.3 Embedded systems	
<input type="checkbox"/> The purpose and characteristics of embedded systems <input type="checkbox"/> Examples of embedded systems	<p>Required</p> <ul style="list-style-type: none"> ✓ What embedded systems are ✓ Typical characteristics of embedded systems ✓ Familiarity with a range of different embedded systems

1.2 – Memory and storage	
Sub topic	Guidance
1.2.1 Primary storage (memory)	
<ul style="list-style-type: none"> <input type="checkbox"/> The need for primary storage <input type="checkbox"/> The difference between RAM and ROM <input type="checkbox"/> The purpose of ROM in a computer system <input type="checkbox"/> The purpose of RAM in a computer system <input type="checkbox"/> Virtual memory 	<p>Required</p> <ul style="list-style-type: none"> ✓ Why computers have primary storage (memory) <ul style="list-style-type: none"> ▪ How this usually consists of RAM and ROM ✓ Key characteristics of RAM and ROM ✓ Why virtual memory may be needed in a system ✓ How virtual memory works <ul style="list-style-type: none"> ▪ Transfer of data between RAM and HDD when RAM is full
1.2.2 Secondary storage	
<ul style="list-style-type: none"> <input type="checkbox"/> The need for secondary storage <input type="checkbox"/> Common types of storage: <ul style="list-style-type: none"> ○ Optical ○ Magnetic ○ Solid state <input type="checkbox"/> Suitable storage devices and storage media for a given application <input type="checkbox"/> The advantages and disadvantages of different storage devices and storage media relating to these characteristics: <ul style="list-style-type: none"> ○ Capacity ○ Speed ○ Portability ○ Durability ○ Reliability ○ Cost 	<p>Required</p> <ul style="list-style-type: none"> ✓ Why computers have secondary storage ✓ Recognise a range of secondary storage devices/media ✓ Differences between each type of storage device/medium ✓ Compare advantages/disadvantages for each storage device ✓ Be able to apply their knowledge in context within scenarios <p>Not required</p> <ul style="list-style-type: none"> ✗ Understanding of the component parts of these types of storage

Sub topic	Guidance
1.2.3 Units	
<ul style="list-style-type: none"> <input type="checkbox"/> The units of data storage: <ul style="list-style-type: none"> ○ Bit ○ Nibble (4 bits) ○ Byte (8 bits) ○ Kilobyte (1,000 bytes or 1 KB) ○ Megabyte (1,000 KB) ○ Gigabyte (1,000 MB) ○ Terabyte (1,000 GB) ○ Petabyte (1,000 TB) <input type="checkbox"/> How data needs to be converted into a binary format to be processed by a computer <input type="checkbox"/> Data capacity and calculation of data capacity requirements 	<p>Required</p> <ul style="list-style-type: none"> ✓ Why data must be stored in binary format ✓ Familiarity with data units and moving between each ✓ Data storage devices have different fixed capacities ✓ Calculate required storage capacity for a given set of files ✓ Calculate file sizes of sound, images and text files <ul style="list-style-type: none"> ▪ sound file size = sample rate x duration (s) x bit depth ▪ image file size = colour depth x image height (px) x image width (px) ▪ text file size = bits per character x number of characters <p>Alternatives</p> <ul style="list-style-type: none"> • Use of 1,024 for conversions and calculations would be acceptable • Allowance for metadata in calculations may be used
1.2.4 Data storage	
<p>Numbers</p> <ul style="list-style-type: none"> <input type="checkbox"/> How to convert positive denary whole numbers to binary numbers (up to and including 8 bits) and vice versa <input type="checkbox"/> How to add two binary integers together (up to and including 8 bits) and explain overflow errors which may occur <input type="checkbox"/> How to convert positive denary whole numbers into 2-digit hexadecimal numbers and vice versa <input type="checkbox"/> How to convert binary integers to their hexadecimal equivalents and vice versa <input type="checkbox"/> Binary shifts 	<p>Required</p> <ul style="list-style-type: none"> ✓ Denary number range 0 – 255 ✓ Hexadecimal range 00 – FF ✓ Binary number range 00000000 – 11111111 ✓ Understanding of the terms ‘most significant bit’, and ‘least significant bit’ ✓ Conversion of any number in these ranges to another number base ✓ Ability to deal with binary numbers containing between 1 and 8 bits <ul style="list-style-type: none"> ▪ e.g. 11010 is the same as 00011010 ✓ Understand the effect of a binary shift (both left or right) on a number ✓ Carry out a binary shift (both left and right)

Sub topic	Guidance
<p>Characters</p> <ul style="list-style-type: none"> <input type="checkbox"/> The use of binary codes to represent characters <input type="checkbox"/> The term ‘character set’ <input type="checkbox"/> The relationship between the number of bits per character in a character set, and the number of characters which can be represented, e.g.: <ul style="list-style-type: none"> ○ ASCII ○ Unicode <p>Images</p> <ul style="list-style-type: none"> <input type="checkbox"/> How an image is represented as a series of pixels, represented in binary <input type="checkbox"/> Metadata <input type="checkbox"/> The effect of colour depth and resolution on: <ul style="list-style-type: none"> ○ The quality of the image ○ The size of an image file <p>Sound</p> <ul style="list-style-type: none"> <input type="checkbox"/> How sound can be sampled and stored in digital form <input type="checkbox"/> The effect of sample rate, duration and bit depth on: <ul style="list-style-type: none"> ○ The playback quality ○ The size of a sound file 	<p>Required</p> <ul style="list-style-type: none"> ✓ How characters are represented in binary ✓ How the number of characters stored is limited by the bits available ✓ The differences between and impact of each character set ✓ Understand how character sets are logically ordered, e.g. the code for ‘B’ will be one more than the code for ‘A’ ✓ Binary representation of ASCII in the exam will use 8 bits <p>Not required</p> <ul style="list-style-type: none"> ✗ Memorisation of character set codes <p>Required</p> <ul style="list-style-type: none"> ✓ Each pixel has a specific colour, represented by a specific code ✓ The effect on image size and quality when changing colour depth and resolution ✓ Metadata stores additional image information (e.g. height, width, etc.) <p>Required</p> <ul style="list-style-type: none"> ✓ Analogue sounds must be stored in binary ✓ Sample rate – measured in Hertz (Hz) ✓ Duration – how many seconds of audio the sound file contains ✓ Bit depth – number of bits available to store each sample (e.g. 16-bit)
1.2.5 Compression	
<ul style="list-style-type: none"> <input type="checkbox"/> The need for compression <input type="checkbox"/> Types of compression: <ul style="list-style-type: none"> ○ Lossy ○ Lossless 	<p>Required</p> <ul style="list-style-type: none"> ✓ Common scenarios where compression may be needed ✓ Advantages and disadvantages of each type of compression ✓ Effects on the file for each type of compression <p>Not required</p> <ul style="list-style-type: none"> ✗ Ability to carry out specific compression algorithms

1.3 – Computer networks, connections and protocols

Sub topic

Guidance

1.3.1 Networks and topologies

<ul style="list-style-type: none"> <input type="checkbox"/> Types of network: <ul style="list-style-type: none"> ○ LAN (Local Area Network) ○ WAN (Wide Area Network) <input type="checkbox"/> Factors that affect the performance of networks <input type="checkbox"/> The different roles of computers in a client-server and a peer-to-peer network <input type="checkbox"/> The hardware needed to connect stand-alone computers into a Local Area Network: <ul style="list-style-type: none"> ○ Wireless access points ○ Routers ○ Switches ○ NIC (Network Interface Controller/Card) ○ Transmission media <input type="checkbox"/> The Internet as a worldwide collection of computer networks: <ul style="list-style-type: none"> ○ DNS (Domain Name Server) ○ Hosting ○ The Cloud ○ Web servers and clients <input type="checkbox"/> Star and Mesh network topologies 	<p>Required</p> <ul style="list-style-type: none"> ✓ The characteristics of LANs and WANs including common examples of each ✓ Understanding of different factors that can affect the performance of a network, e.g.: <ul style="list-style-type: none"> ▪ Number of devices connected ▪ Bandwidth ✓ The tasks performed by each piece of hardware ✓ The concept of the Internet as a network of computer networks ✓ A Domain Name Service (DNS) is made up of multiple Domain Name Servers ✓ A DNS's role in the conversion of a URL to an IP address ✓ Concept of servers providing services (e.g. Web server → Web pages, File server → file storage/retrieval) ✓ Concept of clients requesting/using services from a server ✓ The Cloud: remote service provision (e.g. storage, software, processing) ✓ Advantages and disadvantages of the Cloud ✓ Advantages and disadvantages of the Star and Mesh topologies ✓ Apply understanding of networks to a given scenario
---	--

1.3.2 Wired and wireless networks, protocols and layers

- Modes of connection:
 - Wired
 - Ethernet
 - Wireless
 - Wi-Fi
 - Bluetooth
- Encryption
- IP addressing and MAC addressing
- Standards
- Common protocols including:
 - TCP/IP (Transmission Control Protocol/Internet Protocol)
 - HTTP (Hyper Text Transfer Protocol)
 - HTTPS (Hyper Text Transfer Protocol Secure)
 - FTP (File Transfer Protocol)
 - POP (Post Office Protocol)
 - IMAP (Internet Message Access Protocol)
 - SMTP (Simple Mail Transfer Protocol)
- The concept of layers

Required

- ✓ Compare benefits and drawbacks of wired versus wireless connection
- ✓ Recommend one or more connections for a given scenario
- ✓ The principle of encryption to secure data across network connections
- ✓ IP addressing and the format of an IP address (IPv4 and IPv6)
- ✓ A MAC address is assigned to devices; its use within a network
- ✓ The principle of a standard to provide rules for areas of computing
- ✓ Standards allows hardware/software to interact across different manufacturers/producers
- ✓ The principle of a (communication) protocol as a set of rules for transferring data
- ✓ That different types of protocols are used for different purposes
- ✓ The basic principles of each protocol i.e. its purpose and key features
- ✓ How layers are used in protocols, and the benefits of using layers; for a teaching example, please refer to the 4-layer TCP/IP model

Not required

- ✗ Understand how Ethernet, Wi-Fi and Bluetooth protocols work
- ✗ Understand differences between static and dynamic, or public and private IP addresses
- ✗ Knowledge of individual standards
- ✗ Knowledge of the names and function of each TCP/IP layer

1.4 – Network security

Sub topic	Guidance
1.4.1 Threats to computer systems and networks	
<input type="checkbox"/> Forms of attack: <ul style="list-style-type: none"> ○ Malware ○ Social engineering, e.g. phishing, people as the ‘weak point’ ○ Brute-force attacks ○ Denial of service attacks ○ Data interception and theft ○ The concept of SQL injection 	Required <ul style="list-style-type: none"> ✓ Threats posed to devices/systems ✓ Knowledge/principles of each form of attack including: <ul style="list-style-type: none"> ▪ How the attack is used ▪ The purpose of the attack
1.4.2 Identifying and preventing vulnerabilities	
<input type="checkbox"/> Common prevention methods: <ul style="list-style-type: none"> ○ Penetration testing ○ Anti-malware software ○ Firewalls ○ User access levels ○ Passwords ○ Encryption ○ Physical security 	Required <ul style="list-style-type: none"> ✓ Understanding of how to limit the threats posed in 1.4.1 ✓ Understanding of methods to remove vulnerabilities ✓ Knowledge/principles of each prevention method: <ul style="list-style-type: none"> ▪ What each prevention method may limit/prevent ▪ How it limits the attack

1.5 – Systems software	
Sub topic	Guidance
1.5.1 Operating systems	
<input type="checkbox"/> The purpose and functionality of operating systems: <ul style="list-style-type: none"> ○ User interface ○ Memory management and multitasking ○ Peripheral management and drivers ○ User management ○ File management 	<p>Required</p> <ul style="list-style-type: none"> ✓ What each function of an operating system does ✓ Features of a user interface ✓ Memory management, e.g. the transfer of data between memory, and how this allows for multitasking ✓ Understand that: <ul style="list-style-type: none"> ▪ Data is transferred between devices and the processor ▪ This process needs to be managed ✓ User management functions, e.g.: <ul style="list-style-type: none"> ▪ Allocation of an account ▪ Access rights ▪ Security, etc. ✓ File management, and the key features, e.g.: <ul style="list-style-type: none"> ▪ Naming ▪ Allocating to folders ▪ Moving files ▪ Saving, etc. <p>Not required</p> <ul style="list-style-type: none"> ✗ Understanding of paging or segmentation
1.5.2 Utility software	
<input type="checkbox"/> The purpose and functionality of utility software <input type="checkbox"/> Utility system software: <ul style="list-style-type: none"> ○ Encryption software ○ Defragmentation ○ Data compression 	<p>Required</p> <ul style="list-style-type: none"> ✓ Understand that computers often come with utility software, and how this performs housekeeping tasks ✓ Purpose of encryption, defragmentation and data compression software and why it is required ✓ Utility software is needed to perform additional tasks that may not be carried out by an operating system

1.6 – Ethical, legal, cultural and environmental impacts of digital technology

Sub topic

Guidance

1.6.1 Ethical, legal, cultural and environmental impact

<ul style="list-style-type: none"> □ Impacts of digital technology on wider society including: <ul style="list-style-type: none"> ○ Ethical issues ○ Legal issues ○ Cultural issues ○ Environmental issues ○ Privacy issues □ Legislation relevant to Computer Science: <ul style="list-style-type: none"> ○ The Data Protection Act 2018 ○ Computer Misuse Act 1990 ○ Copyright Designs and Patents Act 1988 ○ Software licences (i.e. open source and proprietary) 	<p>Required</p> <ul style="list-style-type: none"> ✓ Technology introduces ethical, legal, cultural, environmental and privacy issues ✓ Knowledge of a variety of examples of digital technology and how this impacts on society ✓ An ability to discuss the impact of technology based around the issues listed ✓ The purpose of each piece of legislation and the specific actions it allows or prohibits ✓ The need to license software and the purpose of a software licence ✓ Features of open source (providing access to the source code and the ability to change the software) ✓ Features of proprietary (no access to the source code, purchased commonly as off-the-shelf) ✓ Recommend a type of licence for a given scenario including benefits and drawbacks
---	--

2c. Content of Computational thinking, algorithms and programming (J277/02)

2.1 – Algorithms

Sub topic

Guidance

2.1.1 Computational thinking

- Principles of computational thinking:
 - Abstraction
 - Decomposition
 - Algorithmic thinking

Required

- ✓ Understanding of these principles and how they are used to define and refine problems



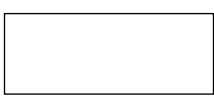
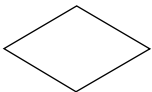


2.1.2 Designing, creating and refining algorithms

- Identify the inputs, processes, and outputs for a problem
- Structure diagrams
- Create, interpret, correct, complete, and refine algorithms using:
 - Pseudocode
 - Flowcharts
 - Reference language/high-level programming language
- Identify common errors
- Trace tables

Required

- ✓ Produce simple diagrams to show:
 - The structure of a problem
 - Subsections and their links to other subsections
- ✓ Complete, write or refine an algorithm using the techniques listed
- ✓ Identify syntax/logic errors in code and suggest fixes
- ✓ Create and use trace tables to follow an algorithm
- ✓ Use of nesting for selection and iteration

Flowchart symbols

	Line		Input/Output
	Process		Decision
	Sub program		Terminal

2.1.3 Searching and sorting algorithms

- Standard searching algorithms:
 - Binary search
 - Linear search
- Standard sorting algorithms:
 - Bubble sort
 - Merge sort
 - Insertion sort

Required

- ✓ Understand the main steps of the algorithm and the segments of code in it
- ✓ Understand any pre-requisites of an algorithm
- ✓ Apply the algorithm to a data set
- ✓ Identify an algorithm if given the code, pseudocode or Exam Reference Language for it

Not required

- ✗ To remember the code, pseudocode or Exam Reference Language for these algorithms

2.2 – Programming fundamentals

Sub topic

Guidance

2.2.1 Programming fundamentals

- The use of variables, constants, operators, inputs, outputs and assignments
- The use of the three basic programming constructs used to control the flow of a program:
 - Sequence
 - Selection
 - Iteration (count- and condition-controlled loops)
- The common arithmetic operators
- The common Boolean operators AND, OR and NOT

Required

- ✓ Practical use of the techniques in a high-level language within the classroom
- ✓ Understanding of each technique
- ✓ Recognise and use the following operators:

Comparison operators		Arithmetic operators	
==	Equal to	+	Addition
!=	Not equal to	–	Subtraction
<	Less than	*	Multiplication
<=	Less than or equal to	/	Division
>	Greater than	MOD	Modulo
>=	Greater than or equal to	DIV	Quotient
		^	Exponentiation (to the power)

2.2.2 Data types

- The use of data types:
 - Integer
 - Real
 - Boolean
 - Character and string
 - Casting

Required

- ✓ Practical use of the data types in a high-level language within the classroom
- ✓ Ability to choose suitable data types for data in a given scenario
- ✓ Understand that data types may be temporarily changed through casting, and where this may be useful

2.2.3 Additional programming techniques

- The use of basic string manipulation
- The use of basic file handling operations:
 - Open
 - Read
 - Write
 - Close
- The use of records to store data
- The use of SQL to search for data
- The use of arrays (or equivalent) when solving problems, including both one-dimensional (1D) and two-dimensional arrays (2D)
- How to use sub programs (functions and procedures) to produce structured code
- Random number generation

Required

- ✓ Practical use of the additional programming techniques in a high-level language within the classroom
- ✓ Ability to manipulate strings, including:
 - Concatenation
 - Slicing
- ✓ Arrays as fixed length or static structures
- ✓ Use of 2D arrays to emulate database tables of a collection of fields, and records
- ✓ The use of functions
- ✓ The use of procedures
- ✓ Where to use functions and procedures effectively
- ✓ The use of the following within functions and procedures:
 - local variables/constants
 - global variables/constants
 - arrays (passing and returning)
- ✓ SQL commands:
 - SELECT
 - FROM
 - WHERE
- ✓ Be able to create and use random numbers in a program

2.3 – Producing robust programs	
Sub topic	Guidance
2.3.1 Defensive design	
<ul style="list-style-type: none"> <input type="checkbox"/> Defensive design considerations: <ul style="list-style-type: none"> ○ Anticipating misuse ○ Authentication <input type="checkbox"/> Input validation <input type="checkbox"/> Maintainability: <ul style="list-style-type: none"> ○ Use of sub programs ○ Naming conventions ○ Indentation ○ Commenting 	<p>Required</p> <ul style="list-style-type: none"> ✓ Understanding of the issues a programmer should consider to ensure that a program caters for all likely input values ✓ Understanding of how to deal with invalid data in a program ✓ Authentication to confirm the identity of a user ✓ Practical experience of designing input validation and simple authentication (e.g. username and password) ✓ Understand why commenting is useful and apply this appropriately
2.3.2 Testing	
<ul style="list-style-type: none"> <input type="checkbox"/> The purpose of testing <input type="checkbox"/> Types of testing: <ul style="list-style-type: none"> ○ Iterative ○ Final/terminal <input type="checkbox"/> Identify syntax and logic errors <input type="checkbox"/> Selecting and using suitable test data: <ul style="list-style-type: none"> ○ Normal ○ Boundary ○ Invalid/Erroneous <input type="checkbox"/> Refining algorithms 	<p>Required</p> <ul style="list-style-type: none"> ✓ The difference between testing modules of a program during development and testing the program at the end of production ✓ Syntax errors as errors which break the grammatical rules of the programming language and stop it from being run/translated ✓ Logic errors as errors which produce unexpected output ✓ Normal test data as data which should be accepted by a program without causing errors ✓ Boundary test data as data of the correct type which is on the very edge of being valid ✓ Invalid test data as data of the correct data type which should be rejected by a computer system ✓ Erroneous test data as data of the incorrect data type which should be rejected by a computer system ✓ Ability to identify suitable test data for a given scenario ✓ Ability to create/complete a test plan

2.4 – Boolean logic

Sub topic

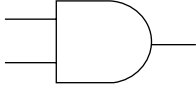
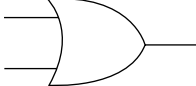
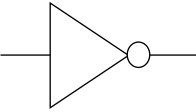
Guidance

2.4.1 Boolean logic

- Simple logic diagrams using the operators AND, OR and NOT
- Truth tables
- Combining Boolean operators using AND, OR and NOT
- Applying logical operators in truth tables to solve problems

Required

- ✓ Knowledge of the truth tables for each logic gate
- ✓ Recognition of each gate symbol
- ✓ Understanding of how to create, complete or edit logic diagrams and truth tables for given scenarios
- ✓ Ability to work with more than one gate in a logic diagram

Boolean Operators	Logic Gate Symbol
AND (Conjunction)	
OR (Disjunction)	
NOT (Negation)	

Truth Tables

AND			OR			NOT	
A	B	A AND B	A	B	A OR B	A	NOT A
0	0	0	0	0	0	0	1
0	1	0	0	1	1	1	0
1	0	0	1	0	1		
1	1	1	1	1	1		

Alternatives

- Use of other valid notation will be accepted within the examination, e.g. Using T/F for 1/0, or V for OR, etc.

2.5 – Programming languages and Integrated Development Environments

Sub topic	Guidance
2.5.1 Languages	
<ul style="list-style-type: none"> <input type="checkbox"/> Characteristics and purpose of different levels of programming language: <ul style="list-style-type: none"> ○ High-level languages ○ Low-level languages <input type="checkbox"/> The purpose of translators <input type="checkbox"/> The characteristics of a compiler and an interpreter 	<p>Required</p> <ul style="list-style-type: none"> ✓ The differences between high-level and low-level programming languages ✓ The need for translators ✓ The differences, benefits and drawbacks of using a compiler or an interpreter <p>Not required</p> <ul style="list-style-type: none"> ✗ Understanding of assemblers
2.5.2 The Integrated Development Environment (IDE)	
<ul style="list-style-type: none"> <input type="checkbox"/> Common tools and facilities available in an Integrated Development Environment (IDE): <ul style="list-style-type: none"> ○ Editors ○ Error diagnostics ○ Run-time environment ○ Translators 	<p>Required</p> <ul style="list-style-type: none"> ✓ Knowledge of the tools that an IDE provides ✓ How each of the tools and facilities listed can be used to help a programmer develop a program ✓ Practical experience of using a range of these tools within at least one IDE

2d. Practical Programming skills

All students must be given the opportunity to undertake a programming task or tasks during their course of study.

The programming task(s) must allow them to develop skills within the following areas when programming:

- Design
- Write
- Test
- Refine

Each task(s) must use one or more high-level text-based programming language, either to a specification or to solve a problem (or problems). They can use any high-level text-based programming language, such as:

- Python
- C family of languages (C#, C++, etc.)
- Java
- JavaScript
- Visual Basic/.Net
- PHP
- Delphi
- BASIC

Some high-level languages do not allow demonstration of all the Practical Programming skills. Where this is the case, schools are encouraged to consider using a second language for practical experience.

Practical Programming skills will be assessed in Component 2 of the qualification, in particular Section B. See Section 3b 'Assessment of Practical Programming skills: Component 2' for more details.



Centres must submit a Practical Programming Statement. See Section 4d for more details.

2e. Prior knowledge, learning and progression

Students in England who are beginning a GCSE (9–1) in Computer Science course are likely to have followed a Key Stage 3 programme of study.

No prior knowledge of this subject is required and there are no prior qualifications required in order for students to enter for a GCSE (9–1) in Computer Science.

GCSEs (9–1) are qualifications that enable students to progress to further qualifications, either Vocational or General.

OCR offer a range of Computing and ICT based qualifications to suit students' needs.

Find out more in Section 7 or at www.ocr.org.uk/computing

3 Assessment of GCSE (9–1) in Computer Science

3a. Forms of assessment

OCR's GCSE (9–1) in Computer Science consists of **two** compulsory components that are externally assessed.

J277/01: Computer systems

This is a compulsory component. It is worth **80 marks**, representing **50%** of the total marks for the GCSE (9–1).

This component is an externally assessed written examination testing AO1 and AO2.

The examination lasts **1 hour 30 minutes**.

All the questions are mandatory.

Students are not permitted to use a calculator in the examination.

The question paper will consist of short and medium answer questions. There will also be one 8-mark extended response question. This question will enable students to demonstrate the ability to construct and develop a sustained line of reasoning.

J277/02: Computational thinking, algorithms and programming

This is a compulsory component. It is worth **80 marks**, representing **50%** of the total marks for the GCSE (9–1).

This component is an externally assessed written examination testing AO1, AO2 and AO3.

The examination lasts **1 hour 30 minutes** and is formed of two sections.

All the questions are mandatory.

Section A is worth **50 marks**, and assesses students' knowledge and understanding of concepts of

Computer Science. Students then apply these to problems in computational terms, where they may use an algorithmic approach.

Section B is worth **30 marks**, and assesses students' Practical Programming skills and their ability to design, write, test and refine programs.

Students are not permitted to use a calculator in the examination.

The question paper will consist of short and medium answer questions.



Sample Assessment Materials and other resources which exemplify our approach to the examinations can be found on the J277 web page of the OCR website.

3b. Assessment of Practical Programming skills: Component 2

All programming code given in examination papers will be presented using the OCR Exam Reference Language.

Section A

Section A assesses a student's ability to structure answers logically without a focus on syntactic precision. Students have flexibility and choice in how they present their answer.

The following table shows how we will set our questions within this section, and how students must respond.

Questions asked in:	Students respond using:
Natural English OCR Exam Reference Language Flowcharts	✓ Pseudocode ✓ Flowcharts ✓ Bullet points ✓ OCR Exam Reference Language or a high-level programming language ✓ Natural English

Section B

Section B assesses a student's ability to design, write, test and refine programs. The following table shows how we will set our questions within this section, and how students must respond.

Question focus	Questions asked in:	Students respond using:
Design	Natural English	✓ Pseudocode ✓ Flowcharts ✓ Tick-box responses ✓ Natural English
Write	Pseudocode Natural English Flowcharts	✓ OCR Exam Reference Language ✓ A high-level programming language
Test	OCR Exam Reference Language	✓ Trace tables ✓ Creating test plans ✓ Identifying suitable test data
Refine	OCR Exam Reference Language	✓ OCR Exam Reference Language ✓ A high-level programming language ✓ Natural English

Where a response requires an answer in OCR Exam Reference Language or a high-level programming language, a student's level of precision will be assessed. These questions are designed to test both a student's programming logic and understanding of core programming structures. Answers written in pseudocode, natural English or bullet points will not be awarded marks.

Responses in OCR Exam Reference Language or a high-level programming language test a student's ability to form an answer using precise programming commands but we will avoid penalising them for minor errors in syntax. This reflects real-life scenarios, where often minor errors would have been flagged within their development environment.

3c. OCR Exam Reference Language

Examination questions will be written in OCR Exam Reference Language for clarity and consistency, apart from ‘Design’ and ‘Write’ questions in Component 2 Section B (please see Section 3b).

Operators							
Comparison operators		Arithmetic operators					
==	Equal to	<=	Less than or equal to	+	Addition	/	Division
!=	Not equal to	>	Greater than	-	Subtraction	MOD	Modulo
<	Less than	>=	Greater than or equal to	*	Multiplication	DIV	Quotient
				^	Exponent		
Boolean operators							
AND	Logical AND						
OR	Logical OR						
NOT	Logical NOT						

Concept	Keyword(s)/Symbols	Example
Commenting		
Comment	//	//This function squares a number function squared(number) squared = number^2 return squared endfunction //End of function
Variables		
Assignment	=	x = 3
Constants	const	name = "Louise" const vat = 0.2
Global Variables	global	global userID = "Cust001"
Input/Output		
Input	input (...)	myName = input("Please enter a name")
Output	print (...)	print("My name is Noni") print(myArray[2,3])
Casting		
Converting to another data type	str()	str(345)
	int()	int("3")
	float()	float("4.52")
	real()	real("4.52")
	bool()	bool("True")

Concept	Keyword(s)/Symbols	Example
Iteration		
FOR loop (Count-controlled)	for ... to ...	for i=0 to 9 print("Loop")
	next ...	next i This will print the word "Loop" 10 times, i.e. 0-9 inclusive .
	for ... to ... step ...	for i=2 to 10 step 2 print(i)
	next ...	next i This will print the even numbers from 2 to 10 inclusive.
		for i=10 to 0 step -1 print(i)
		next i This will print the numbers from 10 to 0 inclusive, i.e. 10, 9, 8,..., 2, 1, 0. Note that the 'step' command can be used to increment or decrement the loop by any positive or negative integer value.
WHILE loop (Condition-controlled)	while ... endwhile	while answer != "Correct" answer = input("New answer") endwhile Will loop until the user inputs the string "Correct". Check condition is carried out before entering loop.
DO UNTIL loop (Condition-controlled)	do until ...	do answer = input("New answer") until answer == "Correct" Will loop until the user inputs the string "Correct". Loop iterates once before a check is carried out.

Concept	Keyword(s)/Symbols	Example
Selection		
IF-THEN-ELSE	<pre>if ... then elseif ... then else endif</pre>	<pre>if answer == "Yes" then print("Correct") elseif answer == "No" then print("Wrong") else print("Error") endif</pre>
CASE SELECT or SWITCH	<pre>switch ... : case ... : case ... : default: endswitch</pre>	<pre>switch day : case "Sat": print("Saturday") case "Sun": print("Sunday") default: print("Weekday") endswitch</pre>

Concept	Keyword(s)/Symbols	Example
String handling/operations		
String length	.length	subject = "ComputerScience" subject.length gives the value 15
Substrings	.substring(x , i) .left(i) .right(i)	subject.substring(3,5) returns "puter" subject.left(4) returns "Comp" subject.right(3) returns "nce" x is starting index; i is number of characters; 0 indexed
Concatenation	+	print(stringA + stringB) print("Hello, your name is: " + name)
Uppercase	.upper	subject.upper gives "COMPUTERSCIENCE"
Lowercase	.lower	subject.lower gives "computerscience"
ASCII Conversion	ASC (...) CHR (...)	ASC('A') returns 65 (numerical) CHR(97) returns 'a' (char)

Concept	Keyword(s)/Symbols	Example
File handling		
Open	<code>open (...)</code>	<code>myFile = open("sample.txt")</code> Note that the file needs to be stored as a variable.
Close	<code>.close()</code>	<code>myFile.close()</code>
Read line	<code>.readLine()</code>	<code>myFile.readLine()</code> returns the next line in the file
Write line	<code>.writeLine(...)</code>	<code>myFile.writeLine("Add new line")</code> Note that the line will be written to the END of the file.
End of file	<code>.endOfFile()</code>	<code>while NOT myFile.endOfFile()</code> <code>print(myFile.readLine())</code> <code>endwhile</code>
Create a new file	<code>newFile()</code>	<code>newFile("myText.txt")</code> Creates a new text file called "myText". The file would then need to be opened using the above command for Open.
Arrays		
Declaration	<code>array colours [...]</code>	<code>array colours[5]</code> Creates 1D array with 5 elements (index 0 to 4). <code>array colours = ["Blue", "Pink", "Green", "Yellow", "Red"]</code> Arrays can be declared with values assigned.
Arrays are 0 indexed Arrays only store a single data type	<code>array gameboard [..., ...] = ...</code>	<code>array gameboard[8, 8]</code> Creates 2D array with 8 elements (index 0 to 7).
Assignment	<code>names [...] = ...</code> <code>gameboard [..., ...] = ...</code>	<code>names[3] = "Noni"</code> <code>gameboard[1, 0] = "Pawn"</code>

Concept	Keyword(s)/Symbols	Example
Sub programs		
Procedure	<pre>procedure name (...)</pre> <pre>endprocedure</pre>	<pre>procedure agePass () print ("You are old enough to ride") endprocedure procedure printName (name) print (name) endprocedure procedure multiply (num1, num2) print (num1 * num2) endprocedure</pre>
Calling a procedure	<pre>procedure (parameters)</pre>	<pre>agePass () printName (parameter) multiply (parameter1, parameter2)</pre>
Function	<pre>function name (...)</pre> <pre>...</pre> <pre>return ...</pre> <pre>endfunction</pre>	<pre>function squared (number) squared = number^2 return squared endfunction</pre>
Calling a function	<pre>function (parameters)</pre>	<pre>print (squared (4)) newValue = squared (4)</pre> <p>Note: Function returns should be stored in a variable if needed for later use in a program.</p>
Random numbers		
Random numbers	<pre>random (... , ...)</pre>	<pre>myVariable = random (1, 6)</pre> <p>Creates a random integer between 1 and 6 inclusive.</p> <pre>myVariable = random (-1.0, 10.0)</pre> <p>Creates a random real number between -1.0 and 10.0 inclusive.</p>

3d. Command words

The command words below will be used consistently in all assessment material and resources.

Command word	Definition
Add	Join something to something else so as to increase the size, number, or amount.
Analyse	Break down in order to bring out the essential elements or structure. Identify parts and relationships, and interpret information to reach conclusions.
Annotate	Add brief notes to a diagram or graph.
Calculate	Obtain a numerical answer showing the relevant stages in the working.
Compare	Give an account of the similarities and differences between two (or more) items or situations, referring to both (all) of them throughout.
Complete	Provide all the necessary or appropriate parts.
Convert	Change the form, character, or function of something.
Define	Give the precise meaning of a word, phrase, concept or physical quantity.
Describe	Give a detailed account or picture of a situation, event, pattern or process.
Design	Produce a plan, simulation or model.
Discuss	Offer a considered and balanced review that includes a range of arguments, factors or hypotheses. Opinions or conclusions should be presented clearly and supported by appropriate evidence.
Draw	Produce (a picture or diagram) by making lines and marks on paper with a pencil, pen, etc.
Evaluate	Assess the implications and limitations. Make judgements about the ideas, works, solutions or methods in relation to selected criteria.
Explain	Give a detailed account including reasons or causes.
Give	Present information which determines the importance of an event or issue, or to show causation.
How	In what way or manner; by what means.
Identify	Provide an answer from a number of possibilities. Recognise and state briefly a distinguishing factor or feature.
Justify	Give valid reasons or evidence to support an answer or conclusion.
Label	Add title, labels or brief explanation(s) to a diagram or graph.
List	Give a sequence of brief answers with no explanation.
Order	Put the responses into a logical sequence.
Outline	Give a brief account or summary.
Refine	Make more efficient, improve, modify or edit.

Command word	Definition
Show	Give steps in a derivation or calculation.
Solve	Obtain the answer(s) using algebraic and/or numerical and/or graphical methods.
State	Give a specific name, value or other brief answer without explanation or calculation.
Tick	Mark (an item) with a tick or select (a box) on a form, questionnaire, etc. to indicate that something has been chosen.
What	Asking for information specifying something.
Write/Rewrite	Mark (letters, words, or other symbols) on a surface, typically paper, with a pen, pencil, or similar implement/write (something) again so as to alter or improve it.

3e. Assessment Objectives

There are three Assessment Objectives (AOs) in OCR GCSE (9–1) in Computer Science. These are detailed in the table below.

Students are expected to:

Assessment Objective	
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: <ul style="list-style-type: none">• to make reasoned judgements• to design, program, evaluate and refine solutions.

Assessment Objective weightings in OCR GCSE (9–1) in Computer Science

The relationship between the Assessment Objectives and the components are shown in the following table:

Component	% of overall GCSE (9–1) in Computer Science (J277)		
	AO1	AO2	AO3
Computer systems (J277/01)	21	29	0
Computational thinking, algorithms and programming (J277/02)	9	11	30
Total (%)	30%	40%	30%

3f. Total qualification time

Total qualification time (TQT) is the total amount of time, in hours, expected to be spent by a learner to achieve a qualification. It includes both guided learning hours and hours spent in preparation, study, and

assessment. The total qualification time for GCSE Computer Science is 140 hours. The total guided learning time is 120-140 hours.

3g. Qualification availability outside of England

This qualification is available in England. For Wales and Northern Ireland please check the Qualifications in Wales Portal (QIW) or the Northern Ireland Department of Education Performance Measures /

Northern Ireland Entitlement Framework Qualifications Accreditation Number (NIEFQAN) list to see current availability.

3h. Language

This qualification is available in English only. All assessment materials are available in English only and all candidate work must be in English.

3i. Assessment availability

There will be one examination series available each year in May/June to **all** students.

This specification will be certificated from the June 2022 examination series onwards.

All examined components must be taken in the same examination series at the end of the course.

3j. Retaking the qualification

Students can retake the qualification as many times as they wish. They must retake all examined components of the qualification.

3k. Assessment of extended response

The assessment materials for this qualification provide students with the opportunity to demonstrate their ability to construct and develop a sustained and coherent line of reasoning.

Marks for extended responses are integrated into the marking criteria for Component 1.

3l. Mathematical skills requirement

In the context of Assessment Objective 2, 'apply' means using knowledge and understanding in a particular context or contexts. It includes both

practical and theoretical contexts, and the use of computing-related mathematics within those contexts.

3m. Synoptic assessment

Synoptic assessment tests students' understanding of the connections between different elements of the subject. It involves the explicit drawing together of knowledge, skills and understanding within different parts of the GCSE (9–1) Computer Science course.

Examination questions in Component 1 and Component 2 will expect students to combine understanding from across the specification in order to provide a full response.

3n. Calculating qualification result

A student's overall qualification grade for GCSE (9–1) in Computer Science will be calculated by adding together their marks from the two written examinations – Component 1 and Component 2.

This mark will then be compared to the qualification-level grade boundaries for the entry option taken by the student, and for the relevant exam series, to determine the student's overall qualification grade.

4 Admin: what you need to know

The information in this section gives an overview of the processes involved in administering this qualification. All of the following processes require you to submit something to OCR by a specific deadline. More information about the processes and deadlines involved at each stage of the assessment cycle can be found in the Administration area of the OCR website.

OCR's *Admin overview* is available on the OCR website at <https://www.ocr.org.uk/administration>

4a. Pre-assessment

Estimated entries

Estimated entries are your best projection of the number of students who will be entered for a qualification in a particular series. Estimated entries

should be submitted to OCR by the specified deadline. They are free and do not commit your centre in any way.

Final entries

Final entries provide OCR with detailed data for each student, showing each assessment to be taken. It is essential that you use the correct entry code, considering the relevant entry rules and ensuring that you choose the entry option for the moderation you intend to use.

Final entries must be submitted to OCR by the published deadlines or late entry fees will apply.

All students taking a GCSE (9–1) in Computer Science must be entered for the following entry option.

Entry option		Components		
Entry code	Title	Code	Title	Assessment type
J277	Computer Science	01	Computer systems	External assessment
		02	Computational thinking, algorithms and programming	External assessment

Collecting evidence of student performance to ensure resilience in the qualifications system

Regulators have published guidance on collecting evidence of student performance as part of long-term contingency arrangements to improve the resilience of the qualifications system. You should review and consider this guidance when delivering this qualification to students at your centre.

For more detailed information on collecting evidence of student performance please visit our website at: <https://www.ocr.org.uk/administration/general-qualifications/assessment/>

4b. Special consideration

Special consideration is a post-assessment adjustment to marks or grades to reflect temporary injury, illness or other indisposition at the time the assessment was taken.

Detailed information about eligibility for special consideration can be found in the JCQ publication *A guide to the special consideration process*.

4c. External assessment arrangements

Regulations governing examination arrangements are contained in the JCQ publication *Instructions for conducting examinations*.

Students are not permitted to use a calculator in either of the externally assessed components.

Head of Centre Annual Declaration

The Head of Centre is required to provide a declaration to the JCQ as part of the annual NCN update, conducted in the autumn term, to confirm that the centre is meeting all of the requirements detailed in the specification.

Any failure by a centre to provide the Head of Centre Annual Declaration will result in your centre status being suspended and could lead to the withdrawal of our approval for you to operate as a centre.

Private candidates

Private candidates may enter for OCR assessments.

A private candidate is someone who pursues a course of study independently but takes an examination or assessment at an approved examination centre. A private candidate may be a part-time student, someone taking a distance learning course, or someone being tutored privately. They must be based in the UK.

Private candidates need to contact OCR approved centres to establish whether they are prepared to host them as a private candidate. The centre may charge for this facility and OCR recommends that the arrangement is made early in the course.

Further guidance for private candidates may be found on the OCR website: <http://www.ocr.org.uk>

4d. Practical Programming skills administration requirements

It is a requirement for all centres to complete and submit a Practical Programming Statement for each year in which students are entered for a GCSE (9–1) in Computer Science.

By signing the statement, your centre is confirming that it has given all students the opportunity to undertake a programming task or tasks during their course of study, as outlined in Section 2d.

The Practical Programming Statement can be downloaded from our website at www.ocr.org.uk

Any failure to submit to OCR a Practical Programming Statement in a timely manner may result in a malpractice/maladministration investigation.

4e. Results and certificates

Grade scale

GCSE (9–1) qualifications are graded on the scale: 9–1, where 9 is the highest. Students who fail to reach the minimum standard of 1 will be Unclassified (U).

Only subjects in which grades 9 to 1 are attained will be recorded on certificates.

Results

Results are released to centres and students for information and to allow any queries to be resolved before certificates are issued.

The following supporting information will be available:

- raw mark grade boundaries for each component.

Centres will have access to the following results information for each student:

- the grade for the qualification
- the raw mark for each component.

Until certificates are issued, results are deemed to be provisional and may be subject to amendment.

A student's final results will be recorded on an OCR certificate. The qualification title will be shown on the certificate as 'OCR Level 1/Level 2 GCSE (9–1) in Computer Science'.

4f. Post-results services

A number of post-results services are available:

- **Review of results** – If you are not happy with the outcome of a student’s results, centres may submit a review of results.
- **Access to scripts** – Centres can request access to marked scripts.
- **Missing and incomplete results** – This service should be used if an individual subject result for a student is missing, or the student has been omitted entirely from the results supplied.

4g. Malpractice

Any breach of the regulations for the conduct of examinations may constitute malpractice (which includes maladministration) and must be reported to OCR as soon as it is detected.

Detailed information on malpractice can be found in the JCQ publication *Suspected Malpractice in Examinations and Assessments: Policies and Procedures*.

5 Appendices

5a. Grade descriptors

Grade 8

To achieve grade 8 candidates will be able to:

- demonstrate relevant and comprehensive knowledge and understanding of fundamental concepts and principles including digital systems and societal impacts
- effectively apply fundamental concepts, principles and mathematical skills, using sustained analytical, logical and evaluative computational thinking, to a wide range of complex problems
- develop and refine a complete solution that meets the requirements of a substantial problem.

Grade 5

To achieve grade 5 candidates will be able to:

- demonstrate mostly accurate and appropriate knowledge and understanding of fundamental concepts and principles including digital systems and societal impacts
- appropriately apply fundamental concepts, principles and mathematical skills, using analytical, logical and evaluative computational thinking, to a range of problems
- produce a working solution that meets most requirements of a substantial problem.

Grade 2

To achieve grade 2 candidates will be able to:

- demonstrate limited knowledge and understanding of fundamental concepts and principles including digital systems and societal impacts
- apply fundamental concepts, principles and mathematical skills, using basic analytical and logical computational thinking, to straightforward problems with limited accuracy
- produce a partially working solution that meets some requirements of a substantial problem.

5b. Overlap with other qualifications

The knowledge, understanding and skills that are developed throughout this qualification are

distinct and have very little overlap with other qualifications.

5c. Accessibility

Reasonable adjustments and access arrangements allow students with special educational needs, disabilities or temporary injuries to access the assessment and show what they know and can do, without changing the demands of the assessment. Applications for these should be made before the examination series.

Detailed information about eligibility for access arrangements can be found in the *JCQ Access Arrangements and Reasonable Adjustments*.

The GCSE (9–1) qualification and subject criteria have been reviewed in order to identify any feature which could disadvantage students who share a protected characteristic as defined by the Equality Act 2010. All reasonable steps have been taken to minimise any such disadvantage.

6 Summary of updates

This specification is an updated version of the J276 GCSE (9–1) Computer Science specification to meet Ofqual requirements introduced for first assessment 2022. All exam boards offering GCSE Computer Science have to assess Practical Programming skills in the external examination from 2022 onwards.



We have summarised the key changes between our J276 and J277 specifications on the next page.

J277: Summary of updates

Date	Version	Section	Title of section	Change
June 2020	2	2b - 1.2.3, 1.2.4, 1.3.1, 1.5.1	Content of Computer systems (J277/01)	Based on feedback received, we have updated our guidance to provide additional support and greater clarity. A black line in the margin indicates where an update has been made.
		2c - 2.1.3, 2.2.3, 2.3.2	Content of Computational thinking, algorithms and programming (J277/02)	
		Exam Reference Language	String handling/operations	Added concatenation
			Arrays	Added an additional way of declaring 1D arrays
		7	Pathways for computing	New section to provide information on other OCR computing qualifications available
Various	Various	Minor updates marked with a black line in the margin		
January 2021	2.1	Cover		Update to specification covers to meet digital accessibility standards
July 2023	2.2	3	Assessment of GCSE (9–1) in Computer Science	Insertion of new section 3f. Total qualification time.

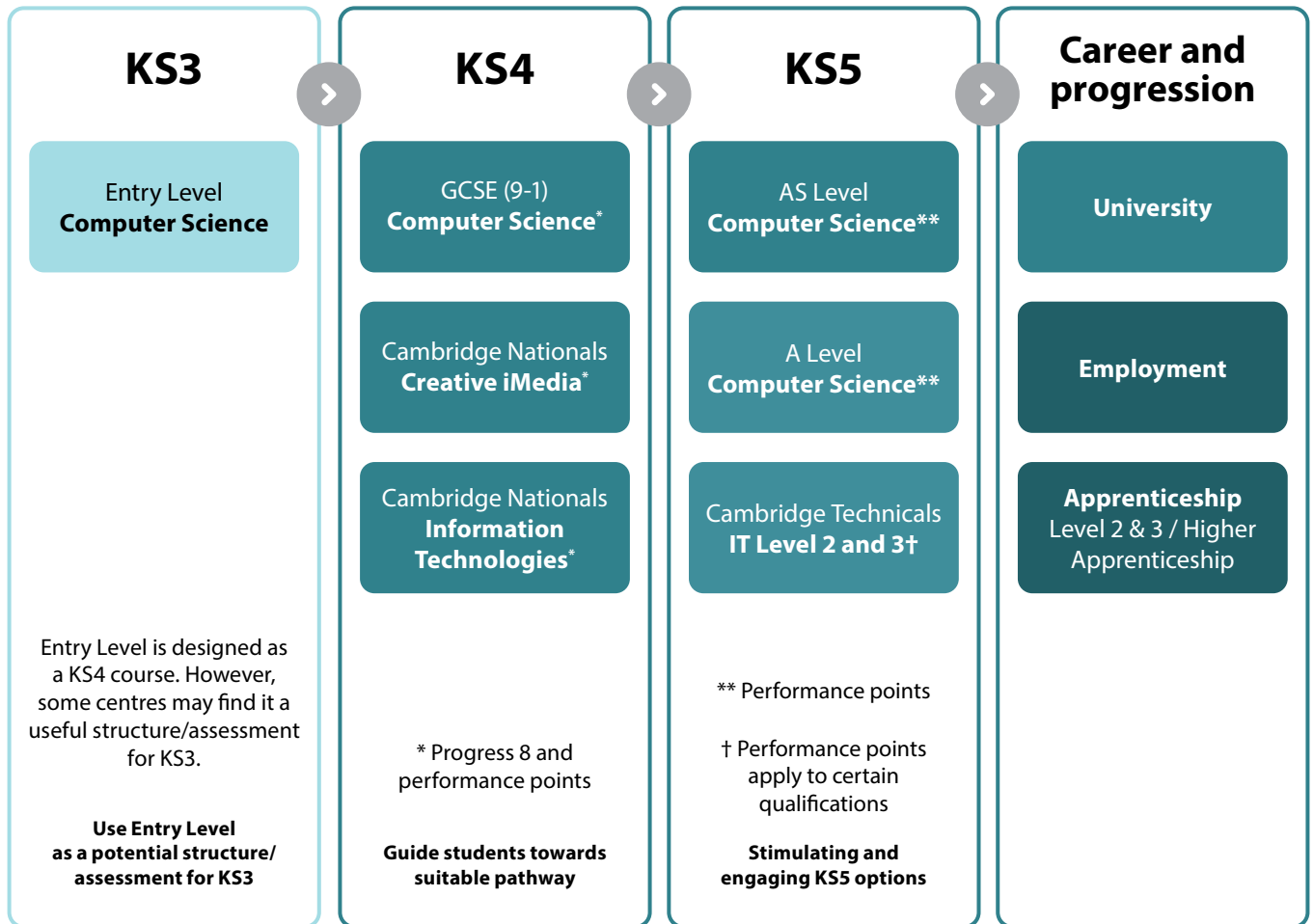
Date	Version	Section	Title of section	Change
February 2024	2.3	2b - 1.5.2	Content of Computer systems (J277/01)	Updated our guidance to provide additional support and greater clarity: In the 'Required' ✓ section: 1.5.2 Clarified the utility software in the second ✓. Added a new ✓ 'Utility software is needed to perform additional tasks that may not be carried out by an operating system'.
		2c - 2.1.2, 2.1.3	Content of Computational thinking, algorithms and programming (J277/02)	2.1.2 Added a new ✓ 'Use of nesting for selection and iteration'. 2.1.3 Updated the first ✓ on algorithms to include 'and the segments of code in it'. Minor grammatical updates to the guidance.
		3c.	OCR Exam Reference Language	Operators: updated Modulus to Modulo. Iteration: updated DO WHILE loop to DO UNTIL loop. String handling: added ' to (A) and corrected formatting of 'a'.
		3g and 3h 4a Checklist	Qualification availability and Language Pre-assessment	Inclusion of disclaimer regarding availability and language. Update to include resilience guidance. Inclusion of Teach Cambridge.

J276 to J277: Summary of updates – key changes

Throughout the process of updating our specification, we engaged with teachers and listened to their feedback. In our survey to teachers and examiners, 100% of respondents said the specification was either extremely or very familiar.

Guidance column added	We have introduced a guidance column in response to customer feedback. Having consulted with teachers, we have made the content tables landscape to make them easier to read.
Content divided into sub-sections	In response to teacher feedback we have divided the content into sub-sections and made sure this was in a logical order for teaching. You may notice the content has moved around in some places.
OCR Pseudocode guide is now the OCR Exam Reference Language	The format of the OCR Exam Reference Language is different to the previous pseudocode guide to make it easier to read, with more examples of its use included.
Moved content out of the appendices	The J276 specification had a lot of information in the appendices. Teachers told us they'd prefer to see this information at the relevant point within the specification e.g. Boolean logic tabulation of symbols.
Data representation moved from Component 2 to Component 1	We have moved data representation content to Component 1 where it is better suited to the theme of the Component. It also allows us to insert the new assessment of programming skills in Component 2 where it fits best.
Practical Programming skills are assessed by examination in Component 2, Section B.	See sections 3a and 3b of the specification for more information.
We have added some content	We have added some content to enable us to assess practical programming skills in the examination, see 2.1, 2.2, 2.3, and 2.5. We have also, for example, refreshed our networks section.
We have updated some content	We have updated content e.g. the Data Protection Act from 1998 to 2018.
We have removed some content	To balance out the additional content, we took the opportunity to review and remove some of our existing content. E.g. packet switching, the concept of virtual networks, characteristics of an assembler, and check digits.

7 Pathways for Computing



KS4 qualifications

We offer a range of qualifications at KS4, each with a different focus. This allows you the ultimate flexibility in how you shape your computing curriculum to suit a wide range of students' needs.

GCSE (9–1) Computer Science	Computer systems, computational thinking, algorithms and programming
Cambridge National Certificate in Information Technologies	IT, data management and project management
Cambridge National Certificate in Creative iMedia	Websites, animation, gaming concepts, sound

YOUR CHECKLIST

Our aim is to provide you with all the information and support you need to deliver our specifications.

- Bookmark [OCR website](#) for all the latest information and news on GCSE (9-1) Computer Science
 - Sign up to [Teach Cambridge](#): our personalised and secure website that provides teachers with access to all planning, teaching and assessment support materials
 - Be among the first to hear about support materials and resources as they become available – register for [GCSE \(9-1\) Computer Science](#)
 - Find out about our [professional development](#)
 - Discover our new online [past paper service](#)
 - Learn more about [Active Results](#)
 - Join our Facebook community at facebook.com/groups/weteachJ277
-

Download high-quality, exciting and innovative GCSE (9-1) Computer Science resources from ocr.org.uk/gcsecomputerscience

Resources and support for our GCSE (9-1) Computer Science qualification, developed through collaboration between our Computer Science Subject Advisors, teachers and other subject experts, are available from our website. You can also contact our Computer Science Subject Advisors who can give you specialist advice, guidance and support.

Contact the team at:

01223 553998

computerscience@ocr.org.uk

@OCR_ict

To stay up to date with all the relevant news about our qualifications, register for email updates at ocr.org.uk/updates

Visit our Online Support Centre at support.ocr.org.uk

follow us on



facebook.com/ocrexams



linkedin.com/company/ocr



@OCRexams



youtube.com/ocrexams



OCR is part of Cambridge University Press & Assessment, a department of the University of Cambridge.

For staff training purposes and as part of our quality assurance programme your call may be recorded or monitored. ©OCR 2024 Oxford Cambridge and RSA Examinations is a Company Limited by Guarantee. Registered in England. Registered office The Triangle Building, Shaftesbury Road, Cambridge, CB2 8EA. Registered company number 3484466. OCR is an exempt charity.

OCR operates academic and vocational qualifications regulated by Ofqual, Qualifications Wales and CCEA as listed in their qualifications registers including A Levels, GCSEs, Cambridge Technicals and Cambridge Nationals.

Cambridge University Press & Assessment is committed to making our documents accessible in accordance with the WCAG 2.1 Standard. We're always looking to improve the accessibility of our documents. If you find any problems or you think we're not meeting accessibility requirements, please [contact us](#).

ocr.org.uk/gcsecomputerscience