Qualification Accredited

OCR
Oxford Cambridge and RSA

GCSE (9–1)
Specification

COMPUTER SCIENCE

J276
For first assessment in 2018

Version 4.0 (August 2018)

ocr.org.uk/gcsecomputerscience

Registered office:
1 Hills Road
Cambridge
CB1 2EU

OCR is an exempt charity.

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.

Contents

Support and Guidance

This specification opens up new opportunities in the subject of computer science. Our aim is to help you at every stage. We work hard with teachers and other experts to bring you a package of practical support, resources and training.

Subject Advisors

OCR Subject Advisors provide information and support to centres including specification updates on resource developments and a range of training opportunities.

Our Subject Advisors work with subject communities through a range of networks to ensure the sharing of ideas and expertise supporting teachers and learners alike.

You can contact our Computer Science Subject Advisors for specialist advice, guidance and support:

01223 553998
ComputerScience@ocr.org.uk
@ocr_ict

Teaching and learning resources

Our resources are designed to provide you with a range of teaching activities and suggestions that enable you to select the best activity, approach or context to support your teaching style and your particular students. We also work with a number of leading publishers who publish textbooks and resources for our specifications. For more information on our publishing partners and their resources visit: ocr.org.uk/qualifications/gcse-and-a-level-reform/publishing-partners

Professional development

Our improved Professional Development Programme fulfils a range of needs through course selection, preparation for teaching, delivery and assessment. Whether you want to come to events, join online training or search for training materials, you can find what you're looking for all in one place at the CPD Hub: cpdhub.ocr.org.uk

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

Assessment Preparation and Analysis Service

Along with subject-specific resources and tools, you'll also have access to a selection of generic resources that focus on skills development, professional guidance for teachers and results data analysis.

Subject Advisor Support

Our Subject Advisors provide you with access to specifications, high-quality teaching resources and assessment materials.

Practice Papers

Assess students' progress under formal examination conditions with question papers downloaded from a secure location, well-presented, easy to interpret mark schemes and commentary on marking and sample answers.

Active Results

Our free online results analysis service helps you review the performance of individual students or your whole cohort. For more details, please refer to ocr.org.uk/activeresults

ExamBuilder

Enabling you to build, mark and assess tests from OCR exam questions and produce a complete mock GCSE or A Level exam. Find out more at ocr.org.uk/exambuilder

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

1a. Why choose an OCR qualification?

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 learners 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 learners 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. We aim to encourage learners to become responsible for their own learning, confident in discussing ideas, innovative and engaged.

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.
- Access to Subject Advisors to support you throughout the lifetime of the specification.
- CPD/Training for teachers to introduce the qualifications and prepare you for first teaching.
- Active Results – our free results analysis service to help you review the performance of individual learners or whole schools.
- ExamBuilder – our new online past papers service that enables you to build your own test papers from past OCR exam questions.

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

Worthwhile

The qualification will build on the knowledge, understanding and skills established through the Computer Science elements of the Key Stage 3 programme of study. The content has been designed not only to allow for a solid basis of understanding but to engage learners and get them thinking about real world application.

Learner-focused

The specification has been developed to improve upon the strengths of OCR's legacy Computing GCSE. The specification will enable learners to develop computational thinking skills built on a sound base of conceptual learning and understanding.

Teacher-centred

OCR offers extensive teacher support material, including resources developed specifically for non-specialists to help ease the transition from ICT to Computer Science. The resources will focus on empowering teachers to explore new teaching methods that will enthuse and engage their learners through the practical application of computational theory.

The Programming Project is designed to be engaging, enabling learners to demonstrate their skills in a way which suits them.

Dependable

OCR's high-quality assessments are backed up by sound educational principles and a belief that the utility, richness and importance of Computer Science should be made evident and accessible to all learners.

OCR's GCSE (9–1) Computer Science specification encourages learners to be inspired, and challenged through completing a coherent, satisfying and worthwhile course of study. The specification will help learners to gain an insight into related sectors. It will prepare learners to make informed decisions about further learning opportunities and career choices.

Aims and learning outcomes

OCR's GCSE (9–1) in Computer Science will encourage learners 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 learners 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 by 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 seamlessly transition into Computer Science at AS Level and/or A Level.

This specification/qualification will enable learners 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?

If you are already using OCR specifications you can contact us at: www.ocr.org.uk

If you are not already a registered OCR centre then you can find out more information on the benefits of becoming one at: www.ocr.org.uk

If you are not yet an approved centre and would like to become one go to: www.ocr.org.uk

Want to find out more?

Ask our Subject Advisors:

Email: computerscience@ocr.org.uk

Teacher support: www.ocr.org.uk

News: @ocr_ict

Customer Contact Centre: 01223 553998

2 The specification overview

2a. OCR's GCSE (9–1) in Computer Science (J276)

Learners take Component 01 and Component 02 to be awarded the OCR GCSE (9–1) in Computer Science.

Content Overview	Assessment Overview	
Computer systems • Systems Architecture • Memory • Storage • Wired and wireless networks • Network topologies, protocols and layers • System security • System software • Ethical, legal, cultural and environmental concerns	Computer systems (01) 80 marks 1 hour and 30 minutes Written paper (no calculators allowed)	50% of total GCSE
Computational thinking, algorithms and programming • Algorithms * • Programming techniques • Producing robust programs • Computational logic • Translators and facilities of languages • Data representation	Computational thinking, algorithms and programming (02) 80 marks 1 hour and 30 minutes Written paper (no calculators allowed)	50% of total GCSE

* Algorithm questions are not exclusive to Component 02 and can be assessed in either component.

Programming Project • Programming techniques • Analysis • Design • Development • Testing and evaluation and conclusions	20 timetabled hours	Formal requirement Consolidates the learning across the specification through practical activity.

2b. Content of Computer systems (J276/01)

This component will introduce learners to the Central Processing Unit (CPU), computer memory and storage, wired and wireless networks, network topologies, system security and system software.
It is expected that learners will become familiar with the impact of Computer Science in a global context through the study of the ethical, legal, cultural and environmental concerns associated with Computer Science.

Learners may draw on some of this content when completing the Programming Project.

1.1 Systems architecture

Learners should have studied the following:

- the purpose of the CPU
- Von Neumann architecture:
  - MAR (Memory Address Register)
  - MDR (Memory Data Register)
  - Program Counter
  - Accumulator
- common CPU components and their function:
  - ALU (Arithmetic Logic Unit)
  - CU (Control Unit)
  - Cache
- the function of the CPU as fetch and execute instructions stored in memory
- how common characteristics of CPUs affect their performance:
  - clock speed
  - cache size
  - number of cores
- embedded systems:
  - purpose of embedded systems
  - examples of embedded sytems.

1.2 Memory

Learners should have studied the following:

- the difference between RAM and ROM
- the purpose of ROM in a computer system
- the purpose of RAM in a computer system
- the need for virtual memory
- flash memory.

1.3 Storage

Learners should have studied the following:

- the need for secondary storage
- data capacity and calculation of data capacity requirements
- common types of storage:
  - optical
  - magnetic
  - solid state
- suitable storage devices and storage media for a given application, and the advantages and disadvantages of these, using characteristics:
  - capacity
  - speed
  - portability
  - durability
  - reliability
  - cost.

1.4 Wired and wireless networks

Learners should have studied the following:

- types of networks:
  - LAN (Local Area Network)
  - WAN (Wide Area Network)
- factors that affect the performance of networks
- the different roles of computers in a client-server and a peer-to-peer network
- the hardware needed to connect stand-alone computers into a Local Area Network:
  - wireless access points
  - routers/switches
  - NIC (Network Interface Controller/Card)
  - transmission media
- the internet as a worldwide collection of computer networks:
  - DNS (Domain Name Server)
  - hosting
  - the cloud
- the concept of virtual networks.

1.5 Network topologies, protocols and layers

Learners should have studied the following:

- star and mesh network topologies
- Wifi:
  - frequency and channels
  - encryption
- ethernet
- the uses of IP addressing, MAC addressing, and 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
- packet switching.

1.6 System security

Learners should have studied the following:

- forms of attack
- threats posed to networks:
  - malware
  - phishing
  - people as the 'weak point' in secure systems (social engineering)
  - brute force attacks
  - denial of service attacks
  - data interception and theft
  - the concept of SQL injection
  - poor network policy
- identifying and preventing vulnerabilities:
  - penetration testing
  - network forensics
  - network policies
  - anti-malware software
  - firewalls
  - user access levels
  - passwords
  - encryption.

1.7 Systems software

Learners should have studied the following:

- the purpose and functionality of systems software
- operating systems:
  - user interface
  - memory management/multitasking
  - peripheral management and drivers
  - user management
  - file management
- utility system software:
  - encryption software
  - defragmentation
  - data compression
  - the role and methods of backup:
    - full
    - incremental.

1.8 Ethical, legal, cultural and environmental concerns

Learners should have studied the following:

- how to investigate and discuss Computer Science technologies while considering:
  - ethical issues
  - legal issues
  - cultural issues
  - environmental issues.
  - privacy issues.
- how key stakeholders are affected by technologies
- environmental impact of Computer Science
- cultural implications of Computer Science
- open source vs proprietary software
- legislation relevant to Computer Science:
  - The Data Protection Act 1998
  - Computer Misuse Act 1990
  - Copyright Designs and Patents Act 1988
  - Creative Commons Licensing
  - Freedom of Information Act 2000.

2c. Content of Computational thinking, algorithms and programming (J276/02)

This component incorporates and builds on the knowledge and understanding gained in Component 01, encouraging learners to apply this knowledge and understanding using computational thinking. Learners will be introduced to algorithms and programming, learning about programming techniques, how to produce robust programs, computational logic, translators and facilities of computing languages and data representation. Learners will become familiar with computing related mathematics.

Learners may draw on some of this content when completing the Programming Project.

2.1 Algorithms

Learners should have studied the following:

- computational thinking:
  - abstraction
  - decomposition
  - algorithmic thinking
- standard searching algorithms:
  - binary search
  - linear search
- standard sorting algorithms:
  - bubble sort
  - merge sort
  - insertion sort
- how to produce algorithms using:
  - pseudocode
  - using flow diagrams
- interpret, correct or complete algorithms.

2.2 Programming techniques

Learners should have studied the following:

- 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 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 and two dimensional arrays
- how to use sub programs (functions and procedures) to produce structured code
- the use of data types:
  - integer
  - real
  - Boolean
  - character and string
  - casting
- the common arithmetic operators
- the common Boolean operators.

2.3 Producing robust programs

Learners should have studied the following:

- defensive design considerations:
  - input sanitisation/validation
  - planning for contingencies
  - anticipating misuse
  - authentication
- maintainability:
  - comments
  - indentation
- the purpose of testing
- types of testing:
  - iterative
  - final/terminal
- how to identify syntax and logic errors
- selecting and using suitable test data.

2.4 Computational logic

Learners should have studied the following:

- why data is represented in computer systems in binary form
- simple logic diagrams using the operations AND, OR and NOT
- truth tables
- combining Boolean operators using AND, OR and NOT to two levels
- applying logical operators in appropriate truth tables to solve problems
- applying computing-related mathematics:
  - +
  - –
  - /
  - *
  - Exponentiation (^)
  - MOD
  - DIV

2.5 Translators and facilities of languages

Learners should have studied the following:

- characteristics and purpose of different levels of programming language, including low level languages
- the purpose of translators
- the characteristics of an assembler, a compiler and an interpreter
- common tools and facilities available in an integrated development environment (IDE):
  - editors
  - error diagnostics
  - run-time environment
  - translators.

2.6 Data representation

Learners should have studied the following:

Units

- bit, nibble, byte, kilobyte, megabyte, gigabyte, terabyte, petabyte
- how data needs to be converted into a binary format to be processed by a computer.

Numbers

- how to convert positive denary whole numbers (0–255) into 8 bit binary numbers and vice versa
- how to add two 8 bit binary integers and explain overflow errors which may occur
- binary shifts
- how to convert positive denary whole numbers (0–255) into 2 digit hexadecimal numbers and vice versa
- how to convert from binary to hexadecimal equivalents and vice versa
- check digits.

Characters

- the use of binary codes to represent characters
- the term 'character-set'
- the relationship between the number of bits per character in a character set and the number of characters which can be represented (for example ASCII, extended ASCII and Unicode).

Images

- how an image is represented as a series of pixels represented in binary
- metadata included in the file
- the effect of colour depth and resolution on the size of an image file.

Sound

- how sound can be sampled and stored in digital form
- how sampling intervals and other factors affect the size of a sound file and the quality of its playback:
  - sample size
  - bit rate
  - sampling frequency.

Compression

- need for compression
- types of compression:
  - lossy
  - lossless.

2d. Content for the Programming Project

The Programming Project provides an opportunity for learners to demonstrate their practical ability in the skills outlined in the specification, supporting the learning of Components 01 and 02. We have provided a cross mapping grid in Section 2f to highlight where Component 01 and 02 may be delivered holistically as part of the Programming Project.

There must be clear evidence submitted to show that each learner has received 20 hours of timetabled delivery for the Programming Project.

The Programming Project requires learners to use skills from Component 01 and Component 02 to create a solution to a set problem. They will code their solution in a suitable programming language. The solution must be tested to ensure they solve the stated problem. Learners must create a suitable test plan with appropriate test data.

The code must be suitably annotated to describe the process. Test results should be annotated to show how these relate to the code, the test plan and the original problem.

Learners will need to provide an evaluation of their solution based on the test evidence.

Learners should be encouraged to be innovative and creative in how they solve the task.

Learners must use a suitable high-level text-based programming language such as:

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

Computational thinking is in essence the ability to model problems in a manner that makes them amenable to computational solutions; it is not simply instructions and actions. Computational thinkers are able to see algorithms, processes and data and know how to then implement them in their chosen language.

When developing a solution to the programming task, we would suggest using an iterative process, such as below:

- **Success criteria** – what key things must the solution contain?
- **Planning and design** – the solution is broken down and suitable designs created
- **Development** – the iterative development with code explanations
- **Testing and remedial actions** – a log of successful tests including correcting any errors
- **Evaluation** – a review of the success criteria that have been met.

This process will allow learners to demonstrate the key elements of computational thinking:

- **Thinking abstractly** – removing unnecessary detail from the problem, and Control and Data abstraction
- **Thinking ahead** – identifying preconditions and inputs and outputs
- **Thinking procedurally** – identifying components of problems and solutions
- **Thinking logically** – predicting and analysing problems
- **Thinking concurrently** – spotting and using similarities.

2e. Programming Project skills

The skills that learners will need for the Programming Project are listed below.

Programming

Learners should have studied the following:

• how to identify and use variables, operators, inputs, outputs and assignments
• how to understand and use the three basic programming constructs used to control the flow of a program: Sequence; Selection; Iteration
• how to understand and use suitable loops including count and condition controlled loops
• how to use different types of data, including Boolean, string, integer and real, appropriately in solutions to problems
• how to understand and use basic string manipulation
• how to understand and use basic file handling operations:
  ○ open
  ○ read
  ○ write
  ○ close
• how to define and use arrays (or equivalent) as appropriate when solving problems
• how to understand and use functions/sub programs to create structured code.

Analysis

Learners should have studied the following:
• how to analyse and identify the requirements for a solution to the problem • how to set clear objectives that show an awareness of the need for real world utility • how to use abstraction and decomposition to design the solution to a problem • how to identify the data requirements for their system • how to identify test procedures to be used during and after development to check their system against the success criteria • how to use validation to ensure a robust solution to a problem.

Design

Learners should have studied the following:
• how to design suitable algorithms to represent the solution to a problem • how to design suitable input and output formats and navigation methods for their system • how to identify suitable variables and structures with appropriate validation for their system • how to use appropriate data types in their system • how to use functions/sub programmes to produce structured reusable code • how to select suitable techniques for the development of the solution.

Development

Learners should have studied the following:
• how to develop a solution to the identified problem using a suitable programming language(s) • how to demonstrate testing and refinement of the code during development • how to explain the solution using suitable annotation and evidence of development • how to use suitable techniques to solve all aspects of the problem • how to take a systematic approach to problem solving • how to deploy practical techniques in an efficient and logical manner • how to show an understanding of the relevant information by presenting evidence of the development of their solutions • how to show an understanding of the technical terminology/concepts that arise from their investigation through analysis of the data collected • how to use the terminology/concepts surrounding their topic and contained in the information collected correctly when it comes to producing analysis in the supporting script.

Testing, evaluation and conclusions

Learners should have studied the following:
• how to produce a full report covering all aspects of the investigation • how to present the information in a clear form which is understandable by a third party and which is easily navigatable • how to critically appraise the evidence that they have presented • how to test their own solution • how to present their evaluation in a relevant, clear, organised, structured and coherent format • how to use specialist terms correctly and appropriately • how to present a conclusion to the report • how to justify their conclusions based on the evidence provided.

2f. Programming Project mapping grid

	Specification Unit	Topic	Formative assessment opportunity
Component 01	1.6 System Security	Threats posed to networks • *Brute force attack* Identifying and preventing vulnerabilities • *Passwords* • *Encryption* • *Network Policies*	Reinforcing ideas about secure network policies, strong passwords and how these relate to brute force attacks (and potential defences against them) should allow candidates to link the importance of these to practical skills in authenticating users.
	1.8 Ethical, legal, cultural and environmental concerns	How to investigate and discuss • *Ethical issues* • *Legal issues* • *Cultural issues* • *Environmental issues* • *Privacy issues*	Some of the Programming Project tasks may cover issues linked to these topics. Wherever students store data, discussions around privacy and legal issues that may arise could be started to reinforce understanding. This may influence how they design their systems. Cultural issues are often an important consideration when designing programs – colour schemes, use of language, character sets, etc. may all be discussed at this point.
		Legislation • *Data Protection Act 1998* • *Computer Misuse Act 1990*	Storage of data and its security can lead to active discussions which may reference the Data Protection Act and Computer Misuse Act. The Programming Project may exemplify some of the scenarios you discuss on this topic. Linking legislation to practical examples can often support students in remembering key issues to discuss in exam questions.

	Specification Unit	Topic	Formative assessment opportunity
Component 02	2.1 Algorithms	Computational Thinking • *Abstraction* • *Decomposition* • *Algorithmic Thinking*	The Programming Project task will require candidates to extract the key elements from the scenario set and to create logical designs – either explicitly through pseudocode or flow charts, or through design when creating code directly. Their ability to do this will reflect how confidently they will be able to read an algorithm question within the examination paper and formulate a suitable response.
		Searching and Sorting Algorithms	Some Programming Project tasks may give candidates the opportunity to see searching and sorting algorithms in operation. Stronger candidates may like the challenge of implementing these searching and sorting algorithms, with the extension activity of explaining their operation to other candidates.
		How to produce algorithms • *Pseudocode* • *Flow charts*	Candidates are expected to create and complete algorithms within the Component 02 examination paper. This is a good time to check how confident they are when creating pseudocode and flow charts within the Programming Project. Although not required, it may be worthwhile to encourage candidates to create both flowcharts and pseudocode for their programs to help embed this process in preparation for the written examination.
		Interpret, correct or complete algorithms	The Programming Project requires candidates to demonstrate these skills. Code reading is a skill in itself and one that requires confidence in understanding of the core concepts of programming techniques. Candidates may be required to read and explain algorithms in the Component 02 examination and this section allows them to demonstrate these skills.

Component 02	2.2 Programming Techniques	All areas	The Programming Project offers candidates a chance to delve into nearly all of the skills listed in this section. For stronger candidates, they should be encouraged to use more advanced skills such as SQL, records and file access. The use of sub programs (functions and procedures) can be used within all of the tasks and will help candidates to learn how modular designs can be implemented, leading to more efficient programs.
	2.3 Producing robust programs	All Areas	Whilst candidates may naturally build defensive design in to their programs, it can help support their development if teachers pose questions to reinforce and/or assess knowledge of this section. It may be appropriate to pose questions such as: • What happens if a user types invalid data into this entry field? • How do you know which parts of the program do what? • How are you making your program more maintainable? • What test data are you using? • Is this test data part of your iterative testing or your final testing?
	2.4 Computational Logic	Truth Tables	Often candidates will want logic statements in their programs. E.g. If it is raining OR it is cold – take coat. Asking candidates to represent their logic choices (especially when complex) through using truth tables can help them understand the logic of their programs and trace why things may not work as expected.
		Applying computer related mathematics	Computing related mathematics may be covered within each Programming Project task. This is a good chance to check understanding of these basic mathematical concepts.

	Specification Unit	Topic	Formative assessment opportunity
Component 02	2.5 Translators and facilities of languages	The IDE – common tools and facilities • *Editors* • *Error Diagnostics* • *Run-time environments* • *Translators*	Candidates may not always use their IDE fully when programming. Before starting the Programming Project, it is an ideal opportunity to cover the key features of the available IDE and show candidates how each feature is used when developing a program. Candidates who can use an IDE confidently will develop more confidence in being able to debug and solve coding issues that may arise.
	2.6 Data Representation	Units	When identifying appropriate data types for data, it could be appropriate to also discuss how much space each value takes in memory when being held in a variable.
		Characters	When manipulating and storing strings, it could be a good time to discuss how strings are collections of characters. This can then be linked into character sets and then into ASCII, Extended ASCII and Unicode.

2g. Prior knowledge, learning and progression

Learners 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 learners to enter for a GCSE (9–1) in Computer Science.

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

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

Find out more at www.ocr.org.uk

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

3a. Forms of assessment

- The GCSE (9–1) in Computer Science is a linear qualification with a 100% terminal rule.
- There are two externally examined components (01 and 02) weighted at 50% each.
- Each examined component consists of an exam paper with a duration of 1 hour 30 minutes.
- Learners must take both components.
- Learners answer all questions in both of the examined components.
- Learners are not permitted to use a calculator in the examinations.
- Some questions will require an extended response. Extended response assessment will enable learners to demonstrate the ability to construct and develop a sustained line of reasoning.

3b. Assessment objectives (AO)

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

Learners 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: • 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 (J276)		
	AO1	AO2	AO3
Computer systems (J276/01)	19	27	4
Computational thinking, algorithms and programming (J276/02)	17	18	15
Total (%)	**36%**	**45%**	**19%**

3c. Assessment availability

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

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

This specification will be certificated from the June 2018 examination series onwards.

3d. Retaking the qualification

Learners can retake the qualification as many times as they wish. They must retake all examined components of the qualification. Learners must either reuse their most recent Programming Project or complete a new Programming Project.

If a learner has previously taken GCSE Computer Science with another awarding body, they may reuse that Programming Project (or equivalent). However, centres must have enough evidence to enable them to submit the Programming Project Authentication Form (CCS161) (See section 4d).

3e. Assessment of extended response

The assessment materials for this qualification provide learners 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 01.

3f. Synoptic assessment

Synoptic assessment tests learners' 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 01 and Component 02 will expect learners to combine understanding from across the specification in order to provide a full response.

3g. Calculating qualification result

A learner'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 01 and Component 02. This mark will then be compared to the qualification level grade boundaries for the entry option taken by the learner, and for the relevant exam series, to determine the learner's overall qualification grade.

The Programming Project does not contribute to the final overall qualification grade.

4 Admin: what you need to know

The information in this section is designed to give 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 http://www.ocr.org.uk/administration

4a. Pre-assessment

Estimated entries

Estimated entries are your best projection of the number of learners 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 learner, 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 learners 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**
J276	Computer Science	01	Computer systems	External assessment
		02	Computational thinking, algorithms and programming	External 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*.

Learners 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. Administration requirements for the Programming Project

All learners are required to complete an individual Programming Project. Although the Programming Project does not count towards a learner's final grade, the skills and knowledge required to complete the Programming Project underpin the content of the examinations.

Learners will choose (or be given) a set Programming Project task from a choice of three tasks that will be provided by OCR. The tasks will be available on the OCR website.

OCR expects teachers to equip learners with the knowledge, understanding and skills before they begin the Programming Project.

Learners will individually produce a report for the Programming Project which demonstrates the content and skills detailed in Sections 2d and 2e.

The report learners write should document their work on the Programming Project task from start to finish. For any code/solution they develop, learners should produce evidence to show they have engaged with each of the sections below:

- Analysis
- Design
- Development
- Testing
- Evaluation

Learners should be supported to reach their own judgments and conclusions when undertaking the task to ensure an authentic experience of creating a coded solution.

OCR has provided a document to help learners structure their Programming Project report. This can be downloaded from the OCR website: www.ocr.org.uk.

Centres must:

- provide learners with 20 timetabled hours to undertake the Programming Project
- ensure the work created is authentic and individual to that learner
- ensure that learners appropriately reference any material used from a source.
- ensure that learners cover each part of the probject

The Programming Project may be completed at any stage of the course. Work must be completed before 15 May in the year that the learner is entered for Component 01 and 02.

Before undertaking the Programming Project, please read our Programming Project Guidance resource. You must use the most recent version that can be found on the OCR website: www.ocr.org.uk

Centres must keep learners work in secure conditions along with related evidence for an appropriate length of time should they wish to re-use the Programming Project as part of retaking the qualification (see 3d).

Each candidate must also complete a Candidate Authentication Statement which confirms the work they have produced is their own. Completed statements are held within your centre.

Monitoring

OCR will contact all centres to request a sample of work for each examination series in which entries are made to ensure the Programming Project requirements have been met.

The sample of work must be submitted to OCR via post (on a USB drive or CD/DVD).

Centres must submit a Programming Project Authentication Form (CCS161) with the sample, confirming that the Programming Project has been completed by all learners in line with the administration requirements. This can be found on the OCR website: www.ocr.org.uk

Evidence of the planning and delivery of the 20 timetabled hours must be submitted along with your centre sample and signed CCS161 form. All evidence submitted must relate to the candidates being entered for that examination series.

Failure to fulfill the Programming Project requirements may result in a malpractice/ maladministration investigation.

4

4e. Results and certificates

Grade Scale

GCSE (9–1) qualifications are graded on the scale: 9–1, where 9 is the highest. Learners 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 learners for information and to allow any queries to be resolved before certificates are issued.

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

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

The following supporting information will be available:

- raw mark grade boundaries for each component.

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

A learner'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:

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

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

1. Grade 8

1.1 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.

2. Grade 5

2.1 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.

3. Grade 2

3.1 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 learners 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 learners who share a protected characteristic as defined by the Equality Act 2010. All reasonable steps have been taken to minimise any such disadvantage.

5d. 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.

5e. Command words

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

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. To identify parts and relationships, and to 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; to 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. Quite often used 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.

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.

5f. Pseudocode, Boolean logic and flowcharts

The following guide shows the format pseudocode will appear in the examined components. It is provided to enable teachers to provide learners with familiarity before the exam. Learners are not expected to memorise the syntax of this pseudocode and, when asked, may provide answers in any style of pseudocode they choose providing its meaning could be reasonably inferred by a competent programmer.

The guide below shows languages and Boolean logic that will be used in the external assessments and indicates the limits and scope of each.

Centres are free to go beyond these parameters.

Variables and constants
Variables and constants are assigned using the = operator.

```
x=3
name="Bob"
```

Variables and constants are declared the first time a value is assigned. They assume the data type of the value they are given.

Variables and constants that are declared inside a function or procedure are local to that subroutine.

Variables in the main program can be made global with the keyword `global`.

```
global userid = 123
```

Variables in the main program can be made constant with the keyword `const`.

```
const vat = 20
```

5

Casting
Variables can be typecast using the `int` `str` and `float` functions.

`str(3)` returns "3"
`int("3")` returns 3
`float("3.14")` returns 3.14

Outputting to screen

```
print(string)
print(variable)
```

Example

```
print("hello")
print(myAge)
```

Taking Input from User
variable=`input(`*prompt to user*`)`

Example

```
name=input("Please enter your name")
```

Iteration – count-controlled

```
for i=0 to 7
    print("Hello")
next i
```

Will print hello 8 times (0-7 inclusive).

Iteration – condition-controlled

```
while answer!=”computer”
    answer=input(“What is the password?”)
endwhile

do
    answer=input(“What is the password?”)
until answer==“computer”
```

Logical operators

```
AND OR NOT

e.g.
while x<=5 AND flag==false
```

Comparison operators

==	Equal to
!=	Not equal to
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to

Arithmetic operators

+	Addition e.g. x=6+5 gives 11
–	Subtraction e.g. x=6–5 gives 1
*	Multiplication e.g. x=12*2 gives 24
/	Division e.g. x=12/2 gives 6
MOD	Modulus e.g. 12MOD5 gives 2
DIV	Quotient e.g. 17DIV5 gives 3
^	Exponentiation e.g. 3^4 gives 81

Selection

Selection will be carried out with if/else and switch/case.

if/else

```
if entry==“a” then
    print(“You selected A”)
elseif entry==“b” then
    print(“You selected B”)
```

```
else
    print("Unrecognised selection")
endif
```

switch/case

```
switch entry:
      case "A":
         print("You selected A")
      case "B":
         print("You selected B")
      default:
         print("Unrecognised selection")

endswitch
```

String handling

To get the length of a string:

```
stringname.length
```

To get a substring:

```
stringname.subString(startingPosition, numberOfCharacters)
```

NB The string will start with the 0th character.

Converting cases:

```
stringname.upper
stringname.lower
```

Ascii conversion:

```
ASC(character)
CHR(asciinumber)
```

Example

```
someText="Computer Science"
print(someText.length)
print(someText.substring(3,3))
```

Will display

```
16
put
```

Subroutines

```
function triple(number)
    return number*3
endfunction
```

Called from main program

```
y=triple(7)

procedure greeting(name)
    print("hello"+name)
endprocedure
```

Called from main program

```
greeting("Hamish")
```

Arrays

Arrays will be 0 based and declared with the keyword *array*.

```
array names[5]
names[0]="Ahmad"
names[1]="Ben"
names[2]="Catherine"
names[3]="Dana"
names[4]="Elijah"

print(names[3])
```

Example of 2D array:

```
array board[8,8]
board[0,0]="rook"
```

Reading to and writing from files

To open a file to read from openRead is used and readLine to return a line of text from the file.

The following program makes x the first line of sample.txt

```
myFile = openRead("sample.txt")
x = myFile.readLine()
myFile.close()
```

endOfFile() is used to determine the end of the file. The following program will print out the contents of sample.txt

```
myFile = openRead("sample.txt")
while NOT myFile.endOfFile()
    print(myFile.readLine())
endwhile
myFile.close()
```

To open a file to write to, openWrite is used and writeLine to add a line of text to the file. In the program below, `hello world` is made the contents of sample.txt (any previous contents are overwritten).

```
myFile = openWrite("sample.txt")
myFile.writeLine("Hello World")
myFile.close()
```

Comments
Comments are denoted by //

```
print("Hello World") //This is a comment
```

Structured Query Language (SQL)

SELECT

FROM

WHERE

LIKE

AND

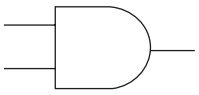
OR

WILDCARDS (*Learners should be familiar in the use of '*' and '%' as a wildcard to facilitate searching and matching where appropriate*).

Boolean algebra

When Boolean algebra is used in questions, the notation described below will be used.

AND – Conjunction



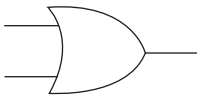
Notation used
∧ e.g. A ∧ B

A	B	A ∧ B
T	T	T
T	F	F
F	T	F
F	F	F

Alternatives accepted:

AND e.g. A AND B
e.g. A

OR - Disjunction



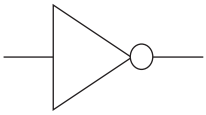
Notation used:
∨ e.g. A∨B

A	B	A∨B
T	T	T
T	F	T
F	T	T
F	F	F

Alternatives accepted:

OR e.g. A OR B
+ e.g. A+B

NOT - Negation



Notation used:
¬ e.g. ¬A

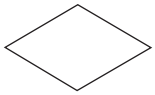
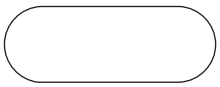
A	¬A
T	F
F	T

Alternatives Accepted:
bar e.g. $\overline{A}$
~ e.g. ~A
NOT e.g. NOT A

5

Flow charts

Flow charts like pseudocode are informal but the most common flow chart shapes are:

	Line	An arrow represents control passing between the connected shapes.
	**Process**	This shape represents something being performed or done.
	**Sub Routine**	This shape represents a subroutine call that will relate to a separate, non-linked flow chart
	**Input/Output**	This shape represents the input or output of something into or out of the flow chart.
	**Decision**	This shape represents a decision (Yes/No or True/False) that results in two lines representing the different possible outcomes.
	**Terminal**	This shape represents the “Start” and “End” of the process.

Summary of updates

Date	Version	Section	Title of section	Change
December 2017	2	Multiple		Changes to generic wording and OCR website links throughout the specification. No changes have been made to any assessment requirements.
February 2018	3	Multiple		In response to Ofqual's decision to change the assessment arrangements, making non-exam assessment no longer count towards a student's 9 to 1 grade in GCSE computer science, the NEA component 03/04 has been removed and has been replaced by the required Programming Project.
April 2018	3.1	i) Front Cover ii) 4e	i) Disclaimer ii) Results and certificates: Results	i) Addition of Disclaimer ii) Amend to Certification Titling
August 2018	3.2	3d	Retaking the qualification	Change to wording relating to 'carry forward' for Programming Project
August 2018	4	Multiple		Added Programming Project cross mapping grid Update and extra clarity to Programming Project requirements Update and extra clarity to Programming Project Administration

YOUR CHECKLIST

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

- ☐ Bookmark **ocr.org.uk/gcsecomputerscience** for all the latest resources, information and news on GCSE (9–1) Computer Science
- ☐ Be among the first to hear about support materials and resources as they become available – register for Computer Science updates at **ocr.org.uk/updates**
- ☐ Find out about our professional development at **cpdhub.ocr.org.uk**
- ☐ View our range of skills guides for use across subjects and qualifications at **ocr.org.uk/skillsguides**
- ☐ Discover our new online past paper service at **ocr.org.uk/exambuilder**
- ☐ Learn more about Active Results at **ocr.org.uk/activeresults**
- ☐ Join our Computer Science social network community for teachers at **social.ocr.org.uk**

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

Computer Science Community

The social network is a free platform where teachers can engage with each other – and with us – to find and offer guidance, discover and share ideas, best practice and a range of Computer Science support materials. To sign up, go to social.ocr.org.uk

follow us on

facebook.com/ocrexams
linkedin.com/company/ocr
@OCR_ICT
youtube.com/ocrexams

ocr.org.uk/gcsecomputerscience

6191854580