

OCR Level 1/2 GCSE (9–1) in Computer Science (J276)

Specification

Version 1: First assessment 2018

This draft qualification has not yet been accredited by Ofqual. It is published to enable teachers to have early sight of our proposed approach to GCSE (9–1) in Computer Science. Further changes may be required and no assurance can be given at this time that the proposed qualification will be made available in its current form, or that it will be accredited in time for first teaching in 2016 and first award in 2018.

Draft

Contents

1	Why choose an OCR GCSE (9–1) in Computer Science?	3
1a.	Why choose an OCR qualification?	3
1b.	Why choose an OCR GCSE (9–1) in Computer Science?	4
1c.	What are the key features of this specification?	5
1d.	How do I find out more information?	5
2	The specification overview	6
2a.	OCR's GCSE (9–1) in Computer Science (J276)	6
2b.	Content of Computer systems (J276/01)	7
2c.	Content of Computational thinking, algorithms and programming (J276/02)	11
2d.	Content for the non-exam assessment (NEA) Programming Project (J276/03/04)	15
2e.	Prior knowledge, learning and progression	19
3	Assessment of GCSE (9–1) in Computer Science	20
3a.	Forms of assessment	20
3b.	Assessment objectives (AO)	20
3c.	Assessment availability	21
3d.	Retaking the qualification	21
3e.	Assessment of extended response	21
3f.	Non-exam assessment	22
3g.	Synoptic assessment	33
3h.	Calculating qualification result	33
4	Admin: what you need to know	34
4a.	Pre-assessment	34
4b.	Special consideration	35
4c.	External assessment arrangements	35
4d.	Admin of non-exam assessment	35
4e.	Results and certificates	36
4f.	Post-results services	37
4g.	Malpractice	37
5	Appendices	38
5a.	Grade descriptors	38
5b.	Overlap with other qualifications	38
5c.	Accessibility	38
5d.	Mathematical skills requirement	38
5e.	Command words	39
5f.	Pseudocode, Boolean logic and flowcharts	40

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 new OCR GCSE (9–1) in Computer Science course 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, Cambridge Technicals and Cambridge Progression.

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 including:
 - Lesson plans
 - Delivery Guides
 - Schemes of Work
 - Lesson Elements
 - ...and much more.
- Access to subject specialists to support you through the transition and throughout the lifetime of the specification.
- CPD/Training for teachers including face-to-face events 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.
- ExamCreator – our new online past papers service that enables you to build your own test papers from past OCR exam questions.
- Codio – an exciting new cloud based coding and content platform that enables teachers and learners to learn computer science practically and in line with our specification. (www.codio.com)

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 QNXXXXXX.

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

Teacher-centred

OCR will offer 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 non-examined assessment 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) specification in Computer Science 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 the theory of Computer Science and one with a focus on programming 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 Specialists 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 specialists:

Email: computerscience@ocr.org.uk

Teacher support: www.ocr.org.uk

News: [@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 (J276)

Learners take Components: 01, 02, and 03; or 01, 02 and 04 to be awarded the OCR GCSE (9–1) in Computer Science.

Content Overview	Assessment Overview	
Computer systems <ul style="list-style-type: none"> • 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)	40% of total GCSE
Computational thinking, algorithms and programming <ul style="list-style-type: none"> • 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)	40% of total GCSE
Programming project ** <ul style="list-style-type: none"> • Programming techniques • Analysis • Design • Development • Testing and evaluation and conclusions 	Programming project (03/04) 40 marks Totalling 20 hours Non-Exam Assessment (NEA)	20% of total GCSE

* Algorithm questions are not exclusive to Component 02 and can be assessed in all components

** Indicates inclusion of synoptic assessment.

Learners who are retaking the qualification may carry forward their result for the non-examined assessment component.

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. It is expected that learners will draw on this underpinning content when completing the Programming Project component (03 or 04).

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 systems.

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.

It is expected that learners will draw on this underpinning content when completing the Programming Project component (03 or 04).

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 non-exam assessment (NEA) Programming Project (J276/03/04)

OCR will issue three assessment tasks at the start of the terminal academic year of assessment. Only tasks designated for that examination series can be submitted unless carrying forward marks from a previous year. The tasks will provide opportunities for the learners to demonstrate their practical ability in the skills outlined in the specification.

Learners will need to create suitable algorithms which will provide a solution to the problems identified in the task. They will then code their solution in a suitable programming language. The solution must be tested at each stage to ensure they solve the stated problem and learners must use 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 approach solving the tasks.

Learners are not allowed access to the internet within the non-exam assessment controlled environment, unless the centre is using an online IDE (Integrated Development Environment). In which case, only access to the IDE website is allowed.

All work submitted by a learner must have been done under observation by their teacher and the final report must be only their own work. External sources can be used but must be referenced and no marks can be awarded for materials submitted which are not the learner's own. Common coded solutions identified as being used by learners will not be given credit during moderation.

Group work can be used to deliver the content and skills but any work submitted as non-exam assessment must be the learner's own.

A form (to be confirmed) will be available at www.ocr.org.uk and will be required upon submission to confirm the validity of the learner's work by the learner, the teacher, and a member of the senior leadership team at the centre.

The non-exam assessment should take a total of 20 hours to complete unless there are specific access requirements that should be considered.

The non-exam assessment should be done using a suitable high level language such as:

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

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.

In Component 03/04 learners must think computationally to solve a task and while doing so create a report detailing the creation of their solution, explaining what they did and why they did it.

The project can be carried out in many ways but is best approached using an iterative process for developing a solution to the task, such as below:

- **Success criteria** (what will a successful solution be)
- **Planning and design** (flow charts and pseudocode)
- **Development** (narrative of the process with explanations of code)
- **Testing and remedial actions** (with narrative of changes made)
- **Evaluation** (clearly linked to success criteria).

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

- **Thinking abstractly** – removing unnecessary detail
- **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.

3.1 Programming techniques

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.

3.2 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.

3.3 Design

Learners should have studied the following:
--

- | |
|--|
| <ul style="list-style-type: none">• 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. |
|--|

3.4 Development

Learners should have studied the following:
--

- | |
|--|
| <ul style="list-style-type: none">• 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. |
|--|

3.5 Testing and evaluation and conclusions

Learners should have studied the following:
--

- | |
|---|
| <ul style="list-style-type: none">• 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. |
|---|

2e. 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.

There are a number of Computing and Computer Science specifications available from OCR.

Find out more at www.ocr.org.uk

Draft

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 three components, two externally examined components (01 and 02) weighted at 40% each and a non-exam assessment (03, 04) weighted at 20% that is assessed by the centre and externally moderated by OCR.
- Each examined component consists of an exam paper with a duration of 1 hour 30 minutes. The non-exam assessment has a duration totalling 20 hours.
- Learners must take all three 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.
- Learners will have a choice of three non-exam assessment tasks to choose from. Learners or centres will need to choose only one task to complete.

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: <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) 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)	15	22	3
Computational thinking, algorithms and programming (J276/02)	14	14	12
Programming project (J276/03/04)	1	4	15
Total (%)	30%	40%	30%

3c. Assessment availability

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

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

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

3d. Retaking the qualification

Learners can retake the qualification as many times as they wish. They can retake all examined components of the qualification. Learners can choose either to retake the non-

exam component or to carry forward their mark for the non-exam component by using the carry forward entry option (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 and Component 03/04.

3f. Non-exam assessment

OCR GCSE (9–1) Computer Science Component 03/04 is the only component which comprises non-examined assessment.

The non-exam assessment in GCSE (9–1) Computer Science will represent 20% of the overall grade for the qualification. The requirements for the subject permit this assessment either to be marked by the awarding organisation, or to be marked by the centre and then moderated by the awarding organisation.

The awarding organisations are working together to agree common approaches for monitoring centres' marking in addition to standard moderation procedures. Specifications will be updated, before teaching of these courses begins, to include information about the processes all awarding organisations will follow.

The learners will choose (or be given) a set task from a choice of three tasks that will be provided by OCR. Learners will produce a report that details the iterative development for the project and demonstrates the following for each task as appropriate.

The methodology is not a formal requirement for the non-exam assessment and is meant for guidance only:

- **Success criteria** (what will a successful solution be)
- **Planning and design** (flow charts and pseudocode)
- **Development** (narrative of the process with explanations of code)

- **Testing and remedial actions** (with narrative of changes made)

- **Evaluation** (clearly linked to success criteria).

Non-exam assessment tasks will be available on OCR's Interchange from **1 September** for that academic year and three new tasks will be produced every year. Guidance on how to access non-exam assessment tasks from Interchange is available on the OCR website. Centres must ensure that learners undertake a task applicable to the correct year of the examination by checking carefully the examination dates of the tasks on Interchange.

Within this specification, OCR expects teachers to equip the learner with the knowledge, understanding and skills before they begin the non-exam assessment task. It should be remembered that learners are required to reach their own judgments and conclusions without any guidance or assistance. When supervising the non-exam assessment task, teachers are expected to offer learners advice on how best to prepare for the research/data collection elements of this component.

Non-exam assessment tasks are available at an early stage to allow planning time. It is anticipated that learners will spend a total of 20 hours in producing the work for this component.

It is essential that any material directly used from a source is appropriately referenced.

Exams directory: www.ocr.org.uk

Programming project

The OCR programming project (Component 03/04) will consist of a task to be solved by the learner. The marking criteria refer to the overall solution set in the project, and for full marks to be available the learner must provide a full solution to the task. The overall solution will be considered against the marking criteria to identify the most appropriate range, and appropriate mark within that range, for each section.

The learners must be formally supervised when working on the non-exam assessment. Formal supervision means under direct teacher supervision: teachers must be able to authenticate the work and there must be acknowledgement and referencing of any sources used.

When supervising tasks, teachers are expected to:

- exercise continuing supervision of work in order to monitor progress and to prevent plagiarism
- ensure that the work is completed in accordance with the specification requirements and can be assessed in accordance with the specified marking criteria and procedures.

Learners must work independently to produce their own final piece of work. It is the responsibility of the Head of Centre to ensure the controls set out in the specification and in the individual components are imposed.

Teachers **may**:

- explain the task
- advise on how the task could be approached
- advise on resources
- alert the learner to key things that must be included in the final piece of work
- interrogate learners to ensure that the work is their own.

Teachers **must not**:

- give detailed advice and suggestions as to how the work may be improved in order to meet the assessment criteria. This includes indicating errors or omissions and personally intervening to improve the presentation or content of the work.
- practise the task with the learners
- practise tasks which are similar in nature with the learners
- provide templates, model answers or feedback on drafts
- produce templates or model answers and publish them online.

Teachers **must** ensure that:

- learners do not access the internet*
- learners are not allowed to take the NEA tasks home with them.
- all work presented for submission must have been completed under supervised conditions.
- accounts associated with the NEA tasks must be locked between sessions to ensure that learners cannot access them outside of the supervised conditions.
- learners do not access online file storage accounts or email during the supervised conditions in order to prevent learners from completing work at home and bringing it into the supervised conditions.

* unless the centre is using an online IDE, in which case, only access to the IDE website is allowed.

Presenting the project

Learners must observe the following procedures when producing their final piece of work for the project tasks:

- any copied material must be suitably acknowledged
- quotations must be clearly marked and a reference provided wherever possible
- work submitted for moderation or marking must be marked with the:
 - centre number
 - centre name
 - learner number
 - learner name
 - component code
 - title assignment title.

- All resources used must be recorded for submission, including but not limited to:

- text books
- teacher resources
- videos
- digital resources

The report should be submitted as a single document (.pdf format preferably) and all the code must be visible in the report and be fully annotated.

Videos can be used where appropriate to evidence work; however, centres must be cautious to not prompt learners for responses as the evidence must be the learners' own.

Work must be submitted in a digital format either online (via the [OCR Repository](#)) or sent via post (on a USB drive or CD/DVD) for moderation in a suitable file structure e.g. a folder for each learner containing the learner's report, the completed URS and signed CCS160_CS forms.

Marking and moderating non-exam assessment

All non-exam assessment components are marked by the centre assessor(s) using OCR marking criteria and guidance, and are moderated by the OCR-appointed moderator.

External moderation is done through e-moderation where evidence in a digital format is supplied via the [OCR Repository](#) or sent via post on a USB drive or CD/DVD.

Applying the marking criteria

The starting point for marking the tasks is the marking criteria. The criteria identify levels of performance for the skills, knowledge and understanding that the learner is required to demonstrate. Before the start of the course,

OCR will provide exemplification through real or simulated learner work which will help to clarify the level of achievement that centre assessors should be looking for when awarding marks.

Use of 'best fit' approach to marking criteria

The assessment task(s) should be marked by teachers according to the marking criteria using a 'best fit' approach. For each of the marking criteria sections, teachers select one of the three band descriptors provided in the marking grid that most closely describes the quality of the work being marked.

Marking should be positive, rewarding achievement rather than penalising failure or omissions. The award of marks **must** be directly related to the marking criteria.

Teachers use their professional judgement in selecting the band descriptor that best describes the work of the learner.

To select the most appropriate mark within the band descriptor, teachers should use the following guidance:

- where the learner's work *convincingly* meets the statement, the highest mark should be awarded
- where the learner's work *adequately* meets the statement, the most appropriate mark in the middle range should be awarded
- where the learner's work *just* meets the statement, the lowest mark should be awarded.

Teachers should use the full range of marks available to them and award full marks in any band for work which fully meets that descriptor. Where there are only two marks within a band the choice will be between work which, in most respects, meets the statement and work which just meets the statement. For wider mark bands, the marks on either side of the middle mark should be used where the standard is lower or higher than 'adequate' but not the highest or lowest mark in the band.

Only one mark per marking criteria section will be entered. The final mark for the learner for each non-exam assessment component is out of a total of 40 marks and is found by totalling the marks for each of the marking criteria sections.

There should be clear evidence that work has been attempted and some work produced. If a learner submits no work for the internally assessed component, then the learner should be indicated as being absent from that component. If a learner completes any work at all for the internally assessed component, then the work should be assessed according to the marking criteria and the appropriate mark awarded, which may be zero.

Annotation of learners' work

Each piece of internally assessed work should show how the marks have been awarded in relation to the marking criteria.

The writing of comments on learners' work, and URS (Unit Recording Sheet), provides a means of communication between teachers

during the internal standardisation and with the moderator if the work forms part of the moderation sample. A URS must be provided for every learner as this ensures the moderator can see how teachers have awarded marks.

Techniques

It is expected that most of the following programming elements and techniques will be used **as appropriate** in the solution of the task. This does not mean that they all have to be identifiably present, but responses that use few of them are unlikely to merit the higher mark bands, this list is not intended to be exclusive. The use of further techniques such as functions might well be appropriate for some learners in order to produce an efficient and elegant solution to a problem.

The mark bands are a guide to the characteristics of work at each level. They are not check lists.

If any component part of the task is not attempted, the final mark must take this omission into account.

Example elements and techniques expected:

- (a) variables, operators, inputs, outputs and assignments
- (b) the three basic programming constructs used to control the flow of a program: sequence; conditionals; iteration
- (c) suitable loops including as appropriate count and condition controlled loops
- (d) different data types including Boolean, string, integer and real appropriately in solutions to problems
- (e) basic string manipulation
- (f) basic file handling operations: open, read, write and close
- (g) arrays or equivalent as appropriate.

Draft

Marking Criteria

Using 'best-fit', decide first which set of marks best describes the overall quality of the answer. Once the band is located, adjust the mark concentrating on features of the answer which make it stronger or weaker following the guidelines for refinement*.

Highest mark: If clear evidence of all the qualities in the band descriptors is shown, the HIGHEST Mark should be awarded.

Lowest mark: If the answer shows the candidate to be borderline (i.e. they have achieved all the qualities of the bands below and show limited evidence of meeting the criteria of the band in question), the LOWEST mark should be awarded.

Middle mark(s): This mark(s) should be used for candidates who are secure in the

band. They are not 'borderline' but they have only achieved some of the qualities in the band descriptors.

Be prepared to use the full range of marks, e.g. do not reserve high band 3 marks 'in case' something turns up of a quality you have not yet seen. If an answer gives clear evidence of the qualities described in the band descriptors, reward appropriately.

*When only two marks are available, only use Highest and Lowest mark guidance for 'best-fit'. When four marks are available, determine if sufficient evidence is available to justify the higher middle mark being awarded, using the band criteria. The teacher commentary on the URS must be used to explain this decision.

Draft

Programming techniques (0 – 12 marks available)

Marks			
Marking criteria	1 – 4	5 – 8	9 – 12
AO3 – 12	<ul style="list-style-type: none"> There is an attempt to program a solution to solve component parts of the task using few of the techniques identified. Code may be minimal, disorganised or hard to follow. Component parts of the task may be trivial, incomplete, or not attempted. 	<ul style="list-style-type: none"> There is an attempt to program a solution to most component parts of the task using several techniques. The code is mostly correct. The task is clearly broken down into its component parts and there are links to the design. A good range of techniques are used appropriately, giving a working solution to most component parts of the task. Some sections of the solution may be inefficiently coded, although basic functionality is mostly successful. The task is mostly complete, but it may be limited in its scope. 	<ul style="list-style-type: none"> There is an attempt to program a solution to solve the task using most of the techniques listed. The task is clearly broken down into its component parts, with reasons given, and there is a clear correspondence of the design with the final code. The techniques are used appropriately in all cases, giving an efficient, working solution to all parts of the problem. There may be additional programming techniques used to produce an efficient and elegant solution to the task. The solution is sufficiently challenging and makes good use of a range of techniques.

0 marks = no response or no response worthy of credit.

Analysis (0 – 6 marks available)

Marks			
Marking criteria	1 – 2	3 – 4	5 – 6
AO2 – 2 AO3 – 4	<ul style="list-style-type: none"> There is limited analysis, with some comments on what the task involves. There may be little decomposition of the task into component parts. There are brief comments on how testing might take place, but with no mention of success criteria. 	<ul style="list-style-type: none"> There is some analysis indicating what is required for each of the component parts. The problems have been decomposed into clearly defined component parts. There is some discussion of how testing will take place. The link between testing and success criteria is evident. In the solution to the task, the objectives are clear. The solution to the task shows awareness of real-world utility value but may be limited in scope. 	<ul style="list-style-type: none"> There is a detailed analysis of what is required for solving the problem. There is clear and logical decomposition of larger tasks into component parts. There is a clear requirements specification which covers all the functionality of the task. Approaches are justified. There is detailed discussion of testing and success criteria. The importance of validation in order to produce a robust program is taken into account. In the solution to the task, objectives are clear and show awareness of the need for real-world utility and robustness.

0 marks = no response or responses not worthy of credit.

Design (0 – 8 marks available)

Marks			
Marking criteria	1 – 2	3 – 5	6 – 8
AO1 – 1 AO2 – 2 AO3 – 5	<ul style="list-style-type: none"> There are basic plans to solve some of the component parts of the task. There may be a limited outline describing the intended approach to some parts of the task. Testing is not planned or only briefly mentioned. The solution may be outlined but not in sufficient detail to produce a workable solution. 	<ul style="list-style-type: none"> Most component parts of the task are planned. The user interface is described. There is a set of basic algorithms outlining a solution to most parts of the problem. There is some discussion of how testing will take place. The link between testing and success criteria is evident. There is some discussion of the variables to be used. There is awareness of the need for validation. The solution to the task shows awareness of real-world utility value, but may be limited in scope. 	<ul style="list-style-type: none"> All the component parts of the task are clearly planned. There is discussion and planning of the user interface. There is a full set of detailed algorithms representing a solution to each part of the problem. The design is complete enough for it to be used as the basis for coding. There is detailed discussion of testing and success criteria. There is awareness of why testing should be destructive. Variables and data structures are identified. The solution to the task shows consideration of building in robustness. The solution to the task is clearly designed in a modular way.

0 marks = no response or responses not worthy of credit.

Development (0 – 8 marks available)

Marks			
Marking criteria	1 – 2	3 – 5	6 – 8
AO1 – 1 AO2 – 2 AO3 – 5	<ul style="list-style-type: none"> There is limited evidence to show a solution to component parts of the task. There is some evidence to show that the solution works. Code is presented with little or no annotation. Variable names may not reflect their purpose. The code shows little organisation or structure. There is a limited record of resources used. Quality of extended response - the report is presented in an unstructured format. Information may be supported by limited evidence and the relationship to the evidence may not be clear. 	<ul style="list-style-type: none"> There is evidence to show how the solution was developed. There is some evidence of testing during development. Testing shows that many parts of the solution work. The code is organised with sensible variable names and with some annotation indicating the purpose of each part of the code. There is a record of resources used. Quality of extended response - the report, for the most part, is presented in a structured format. The information presented is, in the most part, relevant and supported by some evidence. 	<ul style="list-style-type: none"> There is detailed evidence showing development of the solution. There is detailed evidence of systematic testing and refinement during development to show that all parts work as required. The code is well organised with meaningful variable names and detailed annotation indicating the function of each section. There is a detailed record of resources used. Quality of extended response - the report is presented in a coherent and structured format. The information presented is relevant and substantiated.

0 marks = no response or responses not worthy of credit.

Testing and evaluation and conclusions (0 – 6 marks available)

Marks			
Marking criteria	1 – 2	3 – 4	5 – 6
AO2 – 2 AO3 – 4	<ul style="list-style-type: none"> There is evidence to show that the system has been tested for basic functionality but the test plan is limited in scope with little structure. There is limited evidence to show how the result matches the original criteria. Quality of extended response - the report may be ambiguous or disorganised, with little or no use of specialist terms. 	<ul style="list-style-type: none"> There is a test plan covering many parts of the problem with some suggested test data. There is evidence that the system has been tested using this data. There is some evidence to show how the results of testing have been compared to the original criteria. There is a brief evaluation of how successful or otherwise the solution is. Quality of extended response - the report, for the most part, is presented in a structured format with use of some specialist terms. 	<ul style="list-style-type: none"> The detailed test plan covers all major success criteria for the original problem. The testing demonstrates how the solution relates to the success criteria. Unresolved issues are highlighted together with comments on how the issues might be resolved. Testing is clearly more than just a demonstration of success – it is used destructively to uncover errors. There is a full evaluation of the final solution against the success criteria. Quality of extended response - the report is presented in a coherent and structured format, and specialist terms are used accurately and confidently.

0 marks = no response or responses not worthy of credit.

3g. Synoptic assessment

- Synoptic assessment tests the 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.
- The assessment for Component 03/04 will require learners to demonstrate their knowledge of the full content of the specification and is therefore synoptic.

3h. 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 three components taken to give their total weighted mark. 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.

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 these processes, together with the deadlines, can be found in the *OCR Admin Guide and Entry Codes: 14–19 Qualifications*, which can be downloaded from the OCR website: www.ocr.org.uk

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 one of the following entry options:

Entry option		Components		
Entry code	Title	Code	Title	Assessment type
J276 A	Computer Science (OCR Repository)	01	Computer systems	External assessment
		02	Computational thinking, algorithms and programming	External assessment
		03	Programming project	Non-exam assessment (OCR Repository)
J276 B	Computer Science (postal moderation)	01	Computer systems	External assessment
		02	Computational thinking, algorithms and programming	External assessment
		04	Programming project	Non-exam assessment (postal moderation)
J276 C*	Computer Science (carried forward)	01	Computer systems	External assessment
		02	Computational thinking, algorithms and programming	External assessment
		80	Programming project	Non-exam assessment (carried forward)

*Entry option J276 C should only be selected for learners who are retaking the qualification who want to carry forward their mark for the non-exam component.

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 *Instructions for conducting examinations*.

4d. Admin of non-exam assessment

Regulations governing examination arrangements are contained in the JCQ document *Instructions for conducting non-examination assessments*.

OCR's *Admin Guide and Entry Codes: 14–19 Qualifications* provides guidance for centres on the administration of lost or damaged work in centres.

The exam boards and Ofqual are working together to determine the admin guidelines for non-exam assessment in Computer Science GCSE (9–1).

Authentication of learner's work

Learners and centres must declare that the work is the learner's own.

Teachers must declare that the work submitted for internal assessment is the learner's own work by submitting a centre authentication form (CCS160_CS) for each internally assessed component. This form (to

be confirmed) will be available at www.ocr.org.uk and will be required upon submission to confirm the validity of the learner's work by the candidate, the teacher, and a member of the senior leadership team at the centre. This should be sent to the moderator at the same time as the marks.

Internal standardisation

Centres must carry out internal standardisation to ensure that marks awarded by different teachers are accurate

and consistent across all learners entered for the component from that centre.

Moderation

The purpose of moderation is to bring the marking of internally assessed components in all participating centres to an agreed standard. This is achieved by checking a sample of each centre's marking of learners' work.

Following internal standardisation, centres submit marks to OCR and the moderator. If there are fewer than 10 learners, all the work should be submitted for moderation at the same time as marks are submitted.

Once marks have been submitted to OCR and your moderator, centres will receive a moderation sample request. Samples will include work from across the range of attainment of the learners' work.

There are two ways to submit a sample:

Moderation via the OCR Repository –

Where you upload electronic copies of the work included in the sample to the OCR Repository and your moderator accesses the work from there.

Postal moderation – Where you post the sample of work to the moderator.

The method that will be used to submit the moderation sample must be specified when making entries using the codes in Section 4a.

All learners' work must be submitted using the same entry option. It is not possible for centres to offer both options within the same series.

Centres will receive the outcome of moderation when the provisional results are issued. This will include:

Moderation Adjustments Report – Listing any scaling that has been applied to internally assessed components.

Moderator Report to Centres – A brief report by the moderator on the internal assessment of learners' work.

Carrying forward non-exam assessment

Learners who are retaking the qualification can choose either to retake the non-exam assessment or to carry forward their mark for that component from the previous exam series.

If a learner decides to carry forward their mark, they must be entered in the retake series using the entry codes for the carry forward option listed in section 4a.

Learners must decide at the point of entry whether they are going to carry forward the non-exam assessment, or if they are going to retake it to count towards their result.

It is not possible for a learner to retake the non-exam assessment and then choose whether the retake result or a carried forward result is used for certification.

Learners can only carry forward from one year into the following year. Where the gap between the initial qualification and the retake is more than one year, carry forward is not permitted.

A result for a non-exam assessment component can only be carried forward once.

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 total weighted mark for the qualification.

The following supporting information will be available:

- raw mark grade boundaries for each component
- weighted mark grade boundaries for each entry option.

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 1/2 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 and non-exam assessment 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

Ofqual to confirm.

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.

Draft

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
```

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. $12\text{MOD}5$ gives 2
DIV	Quotient e.g. $17\text{DIV}5$ 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(asciinumbr)
```

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)

Learners will be expected to be familiar with the structures below. Should any other aspects of SQL be used, they will be introduced and explained in the question.

`SELECT` *(including nested SELECTs)*

`FROM` *(including use of * wildcard)*

`WHERE`

`LIKE` *(with % used as a wildcard)*

`AND`

`OR`

Boolean algebra

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

AND – Conjunction



Notation used

\wedge e.g. $A \wedge B$

A	B	$A \wedge B$
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:

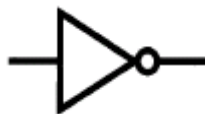
\vee e.g. $A \vee B$

A	B	$A \vee B$
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:

\neg e.g. $\neg A$





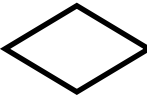

A	$\neg A$
T	F
F	T

Alternatives Accepted:

bar e.g. \overline{A}
~ e.g. $\sim A$
NOT e.g. NOT A

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.