

2 The specification overview

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

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

Content Overview	Assessment Overview
<p>J277/01: Computer systems</p> <p>This component will assess:</p> <ul style="list-style-type: none">• 1.1 Systems architecture• 1.2 Memory and storage• 1.3 Computer networks, connections and protocols• 1.4 Network security• 1.5 Systems software• 1.6 Ethical, legal, cultural and environmental impacts of digital technology	<p>Written paper: 1 hour and 30 minutes 50% of total GCSE 80 marks</p> <p>This is a non-calculator paper.</p> <p>All questions are mandatory.</p> <p>This paper consists of multiple choice questions, short response questions and extended response questions.</p>
<p>J277/02: Computational thinking, algorithms and programming</p> <p>This component will assess:</p> <ul style="list-style-type: none">• 2.1 Algorithms• 2.2 Programming fundamentals• 2.3 Producing robust programs• 2.4 Boolean logic• 2.5 Programming languages and Integrated Development Environments	<p>Written paper: 1 hour and 30 minutes 50% of total GCSE 80 marks</p> <p>This is a non-calculator paper.</p> <p>This paper has two sections: Section A and Section B. Students must answer both sections.</p> <p>All questions are mandatory.</p> <p>In Section B, questions assessing students' ability to write or refine algorithms must be answered using either the OCR Exam Reference Language or the high-level programming language they are familiar with.</p>

2

Practical Programming

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

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



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

1.1 – Systems architecture

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

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

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

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

1.3 – Computer networks, connections and protocols

Sub topic

Guidance

1.3.1 Networks and topologies

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

1.3.2 Wired and wireless networks, protocols and layers

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

Required

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

Not required

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

1.4 – Network security

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

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

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

Sub topic

Guidance

1.6.1 Ethical, legal, cultural and environmental impact

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

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

2.1 – Algorithms

Sub topic

Guidance

2.1.1 Computational thinking

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

Required

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


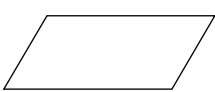

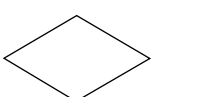


2.1.2 Designing, creating and refining algorithms

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

Required

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

Flowchart symbols

	Line		Input/ Output
	Process		Decision
	Sub program		Terminal

2.1.3 Searching and sorting algorithms

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

Required

- ✓ Understand the main steps of each algorithm
- ✓ Understand any pre-requisites of an algorithm
- ✓ Apply the algorithm to a data set
- ✓ Identify an algorithm if given the code or pseudocode for it

Not required

- ✗ To remember the code for these algorithms
- ✗ To remember Exam Reference Language for Merge Sort

2.2 – Programming fundamentals

Sub topic

Guidance

2.2.1 Programming fundamentals

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

Required

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

Comparison operators

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

Arithmetic operators

+	Addition
–	Subtraction
*	Multiplication
/	Division
MOD	Modulus
DIV	Quotient
^	Exponentiation (to the power)

2.2.2 Data types

☐ The use of data types:

- Integer
- Real
- Boolean
- Character and string
- Casting

Required

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

2.2.3 Additional programming techniques

☐ The use of basic string manipulation

☐ The use of basic file handling operations:

- Open
- Read
- Write
- Close

☐ The use of records to store data

☐ The use of SQL to search for data

☐ The use of arrays (or equivalent) when solving problems, including both one-dimensional (1D) and two-dimensional arrays (2D)

☐ How to use sub programs (functions and procedures) to produce structured code

☐ Random number generation

Required

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

2.3 – Producing robust programs

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

2.4 – Boolean logic

Sub topic



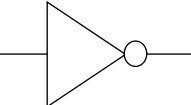
Guidance

2.4.1 Boolean logic

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

Required

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

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

Truth Tables

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

Alternatives

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

2.5 – Programming languages and Integrated Development Environments

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

2d. Practical Programming skills

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

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

- Design
- Write
- Test
- Refine

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

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

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

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



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

2e. Prior knowledge, learning and progression

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

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

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

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

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

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

3a. Forms of assessment

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

J277/01: Computer systems

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

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

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

All the questions are mandatory.

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

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

J277/02: Computational thinking, algorithms and programming

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

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

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

All the questions are mandatory.

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

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

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

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

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



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