GCSE

WJEC GCSE Computer Science

Approved by Qualifications Wales

Specification

Teaching from 2025
For award from 2027

Version 3 - September 2025

This Qualifications Wales regulated qualification is not available to centres in England.

Made for Wales.
Ready for the world.

WJEC would like to thank the students and teachers of Fitzalan High School, Cardiff and Llanwern High School, Newport for their participation in the creation of these covers.

This specification meets the requirements of the following regulatory documents published by Qualifications Wales:

- Made for Wales GCSE Qualification Approval Criteria which set out requirements for any new GCSE qualification Approved for first teaching from September 2025 and beyond.
- Standard Conditions of Recognition which contains the rules that all awarding bodies and their qualifications must meet when offering qualifications to learners in Wales.
- Approval Criteria for GCSE Computer Science which sets out the subject specific requirements for GCSE Computer Science from September 2025 and beyond.

SUMMARY OF AMENDMENTS

Version	Description	Page number
2	Terminal rule change.	30
3	Terminal rule reference removed from 'Post-16 Candidates' section for clarity.	30-31

CONTENTS

GCSE COMPUTER SCIENCE

SUMMARY OF ASSESSMENT

Unit 1: Understanding Computer Science **Digital examination: 1 hour 30 minutes** **50% of qualification** **80 marks**
Questions requiring objective responses, short and extended answers.
Unit 2: Computer Programming **On-screen examination based on a pre-released brief: 2 hours** **50% of qualification** **80 marks**
Set and marked by WJEC. The pre-released brief will include a scenario. A new scenario will be set by WJEC each year. The pre-released brief will be available via the WJEC Portal. The assessment will feature tasks based on the scenario.

This is a unitised qualification.

It is not tiered.

Unit 1 examination will be available in the summer each year. It will be first awarded in 2026.

Unit 2 examination will be available in the summer each year. It will first be awarded in 2027.

It would be most appropriate for learners to complete Unit 2 in the final year of the course. This would allow learners to implement their understanding of computer science concepts developed throughout Unit 1. It would also allow learners to build on the approaches and practices learnt in Unit 1.

The Unit 2 examination will be based on a pre-release brief which will be issued to centres in the academic year before the assessment is scheduled to take place (for example, September 2025 for the 2027 examination). Centres have the flexibility to decide when during the first academic year of study to share the brief with learners.

Qualification Approval Number: C00/4967/3

GCSE COMPUTER SCIENCE

1 INTRODUCTION

1.1 Aims

GCSE Computer Science supports learners to:

- understand and apply principles and concepts of computer science
- explain the features and functions of systems
- develop skills in computational thinking
- model and analyse problems in computational terms in order to develop and implement strategies to solve them
- develop their understanding of, and ability to apply, the systems development life cycle
- apply relevant mathematical skills
- explore relevant social, professional, ethical, environmental and legal dimensions of systems.

These aims are set out in Qualifications Wales' Approval Criteria.

1.2 Curriculum for Wales

This GCSE Computer Science qualification is underpinned by the Curriculum for Wales framework and has been designed to ensure that learners can continue to make progress towards the four purposes whilst studying for this qualification. Central to this design are the principles of progression, along with the statements of what matters and those subject specific skills and concepts outlined in the 'Designing your Curriculum' section of the Science and Technology Area of Learning and Experiences.

In developing this qualification, we have considered where there are opportunities to embed the cross-curricular themes and where there are opportunities for integral skills and cross-curricular skills to be developed. Appendix A provides a simple mapping, and information to support teachers will be provided in the Guidance for Teaching.

We have also considered where the qualification can generate opportunities for integrating the learning experiences noted on page 26; the Guidance for Teaching will include further information on integrating these learning experiences into delivery.

The GCSE Computer Science qualification will support the Curriculum for Wales by:

- supporting the statements of what matters[1] by giving learners the opportunity to:
  - develop and test models
  - consider the social, professional, ethical, environmental and legal impact of systems and the use of technology
  - become enterprising, real-world problem solvers through developing skills in computational thinking.

[1] Science and Technology: Statements of what matters - Hwb (gov.wales)

- supporting the principles of progression[2] by encouraging learners to:
  - engage with iterative problem solving and design
  - explore and experience increasingly complex ideas (for example, in programming tasks in the pre-released brief that are further refined and expanded upon in the examination).

- supporting the subject specific considerations for Computer Science[3] by:
  - learning about the design, development and application of technology, software and systems
  - building understanding of how technologies can impact learners' lives and future careers.

The GCSE Computer Science qualification is based on the following concepts:

- computer architecture
- structure of systems and function
- how systems communicate
- security
- problem solving
  - algorithmic
  - functional decomposition
- programming
- language and compilers
- logical operations
- operating systems
- systems development life cycle.

1.3 Prior learning and progression

Although there is no specific requirement for prior learning, the qualification is designed primarily for learners between the ages of 14 and 16 and builds on the conceptual understanding learners have developed through their learning from ages 3 – 14.

The qualification allows learners to develop a strong foundation of knowledge, skills and understanding which supports progression to post-16 study and prepares learners for life, learning and work. The qualification provides a suitable foundation for the study of Computer Science at either AS or A level. In addition, the specification provides a coherent, satisfying and worthwhile course of study for learners who do not progress to further study in this subject.

1.4 Guided learning hours

GCSE Computer Science has been designed to be delivered within 120 – 140 guided learning hours. The qualification has been primarily designed as a 2-year programme for learners in years 10 and 11.

[2] Science and Technology: Principles of progression - Hwb (gov.wales)
[3] Science and Technology: Statements of what matters - Hwb (gov.wales)

1.5 Use of language

As our understanding of diversity, equity, and inclusion evolves, so must our language. Updated terminology better reflects individual identities and fosters respect and accuracy. Language used should be specific as possible. Staying informed and adaptable is crucial, as inclusive language promotes dignity and equity. Recognising that language will continue to evolve, we will remain open to further amendments to ensure it accurately represents and supports all individuals. WJEC will inform centres of any amendments and the most up to date version of the specification will always be on the website.

1.6 Equality and fair access

The specification may be followed by any learner, irrespective of gender, ethnic, religious or cultural background. It has been designed to avoid, where possible, features that could, without justification, make it more difficult for a learner to access and achieve because they have a particular protected characteristic.

The protected characteristics under the Equality Act 2010 are age, disability, gender reassignment, pregnancy and maternity, race, religion or belief, sex and sexual orientation.

Access arrangements and reasonable adjustments are made for eligible learners to enable them to access the assessments and demonstrate their knowledge and skills, without changing the demands of the assessment.

Information on access arrangements and reasonable adjustments is found in the following document from the Joint Council for Qualifications (JCQ): Access Arrangements, Reasonable Adjustments: General and Vocational Qualifications. This document is available on the JCQ website (www.jcq.org.uk).

We will be following the principles set out in this document and, as a consequence of provision for reasonable adjustments, very few learners will encounter a complete barrier to any part of the assessment.

2 SUBJECT CONTENT

How to read the amplification

The amplification provided in the right-hand column uses the following four stems:

- 'Learners should know' is used when learners are required to use direct recall.
- 'Learners should be aware of' is used when learners do not need to understand all aspects of the specified content in detail. Teachers should refer to Guidance for Teaching documents for further guidance on the depth and breadth to which this content should be taught.
- 'Learners should understand' is used when learners are required to demonstrate greater depth than straight identification or recall. For example, they can apply knowledge to familiar or unfamiliar contexts and can synthesise and evaluate information for a given purpose.
- 'Learners should be able to' is used when learners need to apply their knowledge and understanding to a practical situation or demonstrate application of practical skills and techniques.

The use of the word 'including' indicates that the specified content must be taught and could be subject to assessment.

The use of the words 'for example' or 'such as' indicates that the specified content is for guidance only, and alternative examples could be chosen.

Unit 1

Understanding Computer Science

Digital examination: 1 hour 30 minutes
50% of qualification
80 marks

Overview of unit

The purpose of this unit is to:

- introduce learners to the key concepts and computational processes to be explored throughout the course
- consider the broad legal, social, ethical, environmental and professional consequences relevant to the use of technology
- consider the evolution of technologies that are relevant to the topics.

The unit will be based on the following:

- computer architecture:
  - components
  - peripheral devices
  - storage
- structure of systems and functions:
  - data types, including representation, storage and compression
  - data and file structures
  - automated systems

- how systems communicate:
  - networks and infrastructure
  - cybersecurity and personal privacy
- algorithms
- software:
  - principles of programming
  - software development, including the software development life cycle (SDLC)
  - program construction
- logical operations
- operating systems
- systems development life cycle.

Legal, social, ethical, environmental and professional dimensions and reference to the evolution of technologies will be integrated where appropriate into the above topics.

Areas of content

1.1 Computer architecture	
In this unit learners will develop knowledge, skills and understanding in: 1.1.1 Components 1.1.2 Peripheral devices 1.1.3 Storage	
Content	Further information

1.1.1
Components
	Learners should know:
	• the role of components, including:
	  • motherboard
	  • central processing unit (CPU)
	  • power supply unit (PSU)
	  • input devices
	  • output devices
	  • expansion cards, including:
	  • graphics cards
	  • sound cards
	  • network interface cards (NICs) both wired and wireless
	  • Graphics Processing Unit (GPU)
	  • cooling systems
	  • connectivity ports, including Universal Serial Bus (USB) ports, Display ports, Ethernet ports and audio ports.
	
	Learners should know:
	• the purpose of the CPU
	• the main components of the Von Neumann CPU Architecture
	• the role of different components in the fetch-decode-execute cycle, including:
	  • control unit (CU)
	  • arithmetic logic unit (ALU)
	  • cache memory
	  • registers:
	  • program counter (PC)
	  • current instruction register (CIR)
	  • accumulator (ACC)
	  • memory address register (MAR)
	  • memory data register (MDR).
	
	Learners should know how performance of a CPU is affected by:
	• cache size and levels of cache
	• clock speed
	• number of cores.
	
	Learners should be aware of:
	• the legal, ethical and social consideration surrounding designing, manufacturing and disposing of computer hardware
	• historical development of computers

	• the ethical considerations in designing and manufacturing components • the environmental impact of manufacturing and disposing of hardware • strategies for designing more energy-efficient architectures • historical development of components, from early transistors to modern microprocessors • advancements in materials and manufacturing have influenced the design and efficiency of components over time.
1.1.2 Peripheral devices	Learners should know the function and appropriate use of: • input devices • output devices.

1.1.3
Storage
	Learners should know the use and functional characteristics of primary storage technologies, including:
	• the use and functional characteristics of primary storage technologies, including:
	  • random access memory (RAM)
	  • read only memory (ROM)
	  • flash memory
	  • cache memory
	  • virtual memory.
	
	Learners should know:
	• the use and functional characteristics of secondary storage technologies, including:
	  • optical
	  • magnetic
	  • solid state
	  • cloud storage
	• the general features of secondary storage media in terms of:
	  • capacity
	  • durability
	  • reliability
	  • read/write speed
	  • cost per unit of storage.
	
	Learners should be aware of developments in storage technologies and capacity over time.

1.2 How systems are structured and function

In this unit learners will develop knowledge, skills and understanding in:
1.2.1 Data types, including representation, storage and compression
1.2.2 Data and file structures
1.2.3 Automated systems

Content	Further information

1.2.1
Data types, including representation, storage and compression
	Learners should know:
	• appropriate use of data types
	• the function of data types used in programming
	• the characteristics of different data types, including:
	  • Boolean
	  • character
	  • integer
	  • real
	  • string.
	
	Learners should know how to:
	• convert between, denary, binary and hexadecimal counting systems
	• perform binary addition
	• use arithmetic shift functions and explain their effect
	• use the concept of overflow and underflow.
	
	Learners should understand:
	• the digital storage of characters
	• standardised character sets, including:
	  • Unicode
	  • American Standard Code for Information Interchange (ASCII).
	
	Learners should understand:
	• the digital storage of graphics
	• the effect of different resolutions, bit depths and colour depths on the quality of graphics
	• how to calculate the storage requirements for graphics using different dimensions, bit depths and colour depths.
	
	Learners should understand:
	• the digital storage of sound
	• the effect of different sampling rates on the quality of sound
	• how to calculate the storage requirements for sound using different sampling rates and bit depths.
	
	Learners should understand:
	• the relationship between data storage units using the base 2 prefixes, including:
	  • bit
	  • nibble
	  • byte
	  • kilobyte (i.e. 1,024 bytes)
	  • megabyte (i.e. 1,024 kilobytes)
	  • gigabyte (i.e. 1,024 megabytes)

	• terabyte (i.e. 1,024 gigabytes) • petabyte (i.e. 1,024 terabytes) • the calculation of data capacity requirements. Learners should understand: • the principles of data compression and the use of different compression types, including: • lossy • lossless • calculation of compression ratios and the reduction in file sizes following compression.
1.2.2 Data and file structures	Learners should understand: • the design, interpretation and manipulation of data structures including: • records • one-dimensional arrays • two-dimensional arrays • the selection, identification and justification of appropriate data structures for different situations • the design of files and records appropriate for different applications.
1.2.3 Automated systems	Learners should understand: • the structure of automated systems, including: • sensors • actuators • controllers/processors • communication networks • human-machine interface • the function of automated systems, including: • data acquisition • processing and decision-making • control • communication • feedback and adaptation. Learners should be aware of: • the ethical, social and legal implications of automated systems, such as Artificial Intelligence (AI).

1.3 How systems communicate	
In this unit learners will develop knowledge, skills and understanding in: 1.3.1 Networks and infrastructure 1.3.2 Cybersecurity and personal privacy	
Content	Further information

1.3.1
Networks and infrastructure
	Learners should understand:
	• the characteristics of networks, and their advantages and disadvantages
	• common network topologies, including bus, ring, star and mesh, and their relative advantages and disadvantages
	• the hardware required to establish wired and wireless connectivity, including:
	  • routers
	  • hubs
	  • switches
	  • bridges
	  • gateways
	  • wireless access points (WAPs).
	
	Learners should know the purpose of networking protocols, including:
	• Transmission Control Protocol (TCP)
	• Internet Protocol (IP)
	• Hypertext Transfer Protocol (HTTP)
	• Hypertext Transfer Protocol Secure (HTTPS)
	• File Transfer Protocol (FTP)
	• Simple Mail Transfer Protocol (SMTP)
	• Post Office Protocol (POP3)
	• Internet Message Access Protocol (IMAP)
	• Border Gateway Protocol (BGP)
	• Domain Name System (DNS)
	• packet switching and the typical contents of a TCP / IP packet
	• cost calculation of routing data on a network.
	
	Learners should be aware of:
	• the evolution of communication networks, from traditional telecommunication systems to modern high-speed broadband and networks.

1.3.2 Cybersecurity and personal privacy	Learners should know the characteristics and definitions of threats to computer systems, including: • malware such as viruses, trojan horses, ransomware, worms and spyware • social engineering such as phishing, pretexting, baiting, quid pro quo and impersonation • brute force attacks such as simple brute force, dictionary attacks, hybrid brute force, reverse brute force and credential stuffing • denial of service attacks such as distributed denial-of-service (DDoS), volume-based attacks, protocol attacks and application-layer attacks

	• data interception and theft such as man-in-the-middle (MiTM) attacks, packet sniffing, keylogging, session hijacking and IP spoofing • SQL injection such as error-based, union-based, blind Boolean-based and blind time-based. Learners should be aware of the legal and ethical aspects of network security, including privacy concerns related to data transmission over networks. Learners should know the purpose of different methods of protecting against threats during system design, creation, testing and use, including: • penetration testing • network forensics • anti-malware software • firewalls • user access levels • passwords • backups • Multifactor Authentication (MFA) • secure by design • encryption.

1.4 Algorithms

In this unit learners will develop knowledge, skills and understanding in:
1.4.1 Algorithms

Content	Further information
1.4.1 Algorithms	Learners should know how to create and interpret different methods of defining algorithms, including pseudo-code and flowcharts. Learners should understand how to write, correct, test and interpret the function of algorithms that solve real-life problems using: • self-documenting identifiers • subroutines • input • output • sequence, selection and iteration • counts and rogue values • string handling • mathematical and logical operations • local and global variables • data structures, including arrays up to two-dimensions. Learners should know the characteristics of, and how to perform a search and sort algorithm, including: • linear search • binary search • bubble sort • selection sort • insertion sort. Learners should understand: • the purpose of and techniques for data validation and verification • algorithms that validate and verify data.

1.5 Software	
In this unit learners will develop knowledge, skills and understanding in: 1.5.1 Principles of programming 1.5.2 Software development, including the software development life cycle (SDLC) 1.5.3 Program construction	
Content	Further information
1.5.1 Principles of programming	Learners should know the principles of good programming in terms of: • readability • extensibility • maintainability • modularity.
1.5.2 Software development life cycle (SDLC)	Learners should know the principles of project management, including: • project initiation and planning • project execution and monitoring • project control and evaluation. Learners should understand the stages of software development, including: • planning and analysis: “Investigation” • design • development and implementation • testing • deployment • maintenance, support and refinement. Learners should be aware of the legal, ethical and social considerations surrounding software development.

1.5.3
Program construction
	Learners should understand:
	• the characteristics and purpose of high-level and low-level languages
	• situations that require the use of a high-level or a low-level language
	• the purpose and use of the three common types of program translator:
	  • compilers
	  • interpreters
	  • assemblers.
	
	Learners should understand:
	• the principal stages involved in the compilation process:
	  • lexical analysis
	  • syntax analysis
	  • semantic analysis
	  • code generation
	  • code optimisation
	• examples of programming errors.

1.6 Logical operations	
In this unit learners will develop knowledge, skills and understanding in: 1.6.1 Logical operators	
Content	Further information

1.6.1
Logical operators
	Learners should understand:
	• the use of logical operators in truth tables to solve problems, including:
	  • AND
	  • OR
	  • NOT
	  • XOR.
	
	Learners should be aware of the historical context of logical operations in computing, from early Boolean algebra applications to modern digital circuits.

1.7 Operating systems	
In this unit learners will develop knowledge, skills and understanding in: 1.7.1 Operating systems	
Content	Further information

1.7.1
Operating systems
	Learners should understand:
	• the purpose of the operating system (OS)
	• how the operating system manages:
	  • CPU
	  • multi-tasking
	  • interrupts
	  • memory
	  • backing store
	  • peripherals
	  • interface
	  • security
	• the purpose and functionality of a range of utility software.
	
	Learners should be aware of:
	• legal and ethical considerations related to security features in operating systems
	• user privacy concerns in the context of OS design.

1.8 Systems development life cycle

In this unit learners will develop knowledge, skills and understanding in:
1.8.1 Systems development life cycle (SDLC)

Content	Further information

1.8.1
Systems development life cycle (SDLC)
	Learners should understand:
	• the broader perspective of the development of entire systems, which may include software, hardware, people, processes and data
	• the systematic process and phases followed by developers in the SDLC, including:
	  • scoping and feasibility
	  • analysis
	  • design
	  • implementation
	  • testing
	  • deployment
	  • maintenance
	• the use, strengths and weaknesses of different SDLC models, including:
	  • waterfall
	  • agile
	  • iterative
	  • spiral.
	
	Learners should be aware of:
	• professional considerations related to industry standards for software development
	• legal and ethical considerations relating to intellectual property in software development.

Unit 2

Computer programming

On-screen examination based on a pre-release brief: 2 hours
50% of qualification
80 marks

Overview of unit

The purpose of this unit is to:

- explore the concept of programming
- develop programming skills using Python as the specified language
- encourage iterative problem solving and design
- develop the use of data modelling skills
- give learners the opportunity to build appropriate user interfaces.

The unit will be based on the following:

- investigation
  - decomposition
  - abstraction
  - pattern recognition
- design
  - algorithms
  - data modelling
- implementation
  - programming
  - user interfaces
- testing
- refinement, including evaluation.

Learners will require access to Python 3. The version of Python 3 required will be specified on the scenario issued to candidates and on the examination instructions.

The materials will be released, written using the standard Python IDLE IDE and will use TKINTER library for graphical user elements.

Centres may use an IDE of their choice for teaching programming knowledge and skills. However, learners are required to use the Python IDLE IDE version specified in the pre-release for the Unit 2 examination.

Areas of content

2.1 Investigation	
In this unit learners will gain knowledge and understanding of the following areas: 2.1.1 Investigation Learners should be able to analyse Computer Science problems, using decomposition, abstraction, pattern recognition and algorithms to determine solutions in terms of the following as outlined below:	
Content	Further information
2.1.1 Investigation	• use a systematic approach to problem solving including the use of decomposition and abstraction • use abstraction effectively to model selected aspects of the external world in an algorithm or program • use abstraction effectively to appropriately structure programs into modular parts with clear, well-documented interfaces • use data modelling skills to determine requirements • analyse a set of requirements • consider a programming solution that meets a set of requirements.

2.2 Design

In this unit learners will gain knowledge and understanding of the following areas:
2.2.1 Design

Learners should be able to design solutions and their component parts, and use data modelling skills in terms of the following as outlined below:

Content	Further information

2.2.1
Design
	• design and document the input and output facilities required to produce an effective user interface
	• design and document all required data structures
	• using pseudo code, design and document the following routines:
	  • validation and verification
	  • data handling and processing
	  • authentication
	  • processing
	  • user interface designs
	• create and modify data validation and verification routines.

2.3 Implementation

In this unit learners will gain knowledge and understanding of the following areas:
2.3.1 Implementation

Learners should be able to use the Python programming language to write programs and build appropriate user interfaces in terms of the following as outlined below:

Content	Further information

2.3.1
Implementation
	Design, write, test and refine Python 3 code using the following skills:
	• create new and extend existing functions or methods
	• create new, edit and extend existing Python 3 modules
	• use variables, operators, inputs, outputs and assignment
	• use a variety of data types, including integer, Boolean, real, character and string
	• use programming constructs to control the flow of a program, including:
	  • iteration (conditional and count controlled loops)
	  • selection
	  • sequence
	• use basic file handling, including:
	  • open a file
	  • read from a file into a variable
	  • read from a file into a list (which function similarly to arrays in other programming languages)
	  • write to a file
	  • write to a file from a list (which function similarly to arrays in other programming languages)
	  • close a file
	  • create new and extend data structures and fixed length records to store data
	• use string manipulation
	• create new and extend lists, tuples and dictionaries
	• use lists (which function similarly to arrays in other programming languages)
	• use slicing
	• use mathematical and logical operations
	• create new and modify existing intuitive Graphical User Interfaces (GUI) using the Python 3 built in Tkinter libraries, including:
	  • windows
	  • frames
	  • widgets
	  • buttons
	  • geometry managers
	• create and modify authentication management routines
	• create code for the solution that is self-documenting and uses meaningful identifiers
	• use a programming style that is consistent, including indentation and appropriate use of white space
	• use subroutines with well-defined interfaces
	• annotate code so that it is accessible to a competent third party.

2.4 Testing, refinement, evaluation

In this unit learners will gain knowledge and understanding of the following areas:
2.4.1 Testing
2.4.2 Refinement and evaluation

Learners should be able to use the Python programming language to test and refine effective programs and build appropriate user interfaces in terms of the following:

Content	Further information
2.4.1 Testing	• design and document an effective testing strategy that will ensure that the final solution meets the requirements • describe how the outcomes of a testing process can be used to inform further development of the solution • design test data to include examples of typical, extreme and erroneous content • implement a test plan using typical, extreme and erroneous data where appropriate • present test outcomes with detailed and informed commentaries • demonstrate testing and refinement of code during development or in response to change in the requirements provided • explain the outcome of testing using given data or data that the learner has designed.
2.4.2 Refinement and evaluation	• demonstrate understanding of the requirements and any changes required by revised requirements • demonstrate understanding of changes by presenting evidence of developmental changes • demonstrate testing and refinement of code to improve efficiency • discuss the strengths and weaknesses of differing approaches to meeting the requirements • demonstrate an understanding of the technical terminology/concepts used during software development • evaluate the outcomes of the original given code • evaluate the outcomes of the changes made by additional requirements • explain the solution or changes made to a solution.

Opportunities for integration of learning experiences

GCSE Computer Science generates opportunities for the following learning experiences to be developed (the experiences will not be directly assessed):

- undertake 'unplugged' learning and development activities to help bring to life computational concepts
- engage with examples and achievements in programming that have provided solutions to real-world problems
- engage in discussions about the potential impacts of the latest Artificial Intelligence (AI) tools and technologies, and developments in this field
- use design-thinking to understand how a range of approaches could solve computer science problems
- develop collaboration and teamwork skills by working with, and learning from, others
- gain awareness and appreciation of some of the different careers and work-related areas that draw upon computer science.

The Guidance for Teaching will include further information on the opportunities provided by the qualification, for teachers/centres to integrate these learning experiences into delivery.

For opportunities to develop cross-cutting themes, cross-curricular skills and integral skills please see Appendix A.

3 ASSESSMENT

3.1 Assessment objectives and weightings

Below are the assessment objectives for this specification. Learners must:

AO1
Demonstrate knowledge and understanding of the key concepts and principles of computer science

AO2
Apply knowledge and understanding of key concepts and principles of computer science

AO3
Analyse problems in computational terms:
- to make reasoned judgements
- to design, program, evaluate, and refine solutions.

The table below shows the weighting of each assessment objective for each unit and for the qualification as a whole.

	AO1	AO2	AO3	Total
Unit 1	30%	15%	5%	50%
Unit 2	0%	25%	25%	50%
Overall weighting	**30%**	**40%**	**30%**	**100%**

3.2 Arrangements for pre-release brief and examination for Unit 2

For this assessment it is required that learners are provided with a pre-release brief that includes information for candidates and a scenario. This brief will be made available to centres in September of the first year of study (Year 10) in preparation for the examination in the second year of study (Year 11). The pre-release brief will be made available on the WJEC Portal. It is up to centres to choose exactly when to distribute the pre-release brief to learners, but it MUST be during Year 10.

Learners will be required to familiarise themselves with the scenario ready to complete an examination based on the pre-released brief in the summer examination series or Year 11. It must be emphasised that learners are not permitted to take any prompts or notes into the examination. Learners will be provided with a clean copy of the pre-release materials when sitting the examination.

This assessment contributes to 50% of the overall qualification grade and will take 2 hours to complete. The assignment will be marked out of a total of 80 marks.

4 MALPRACTICE

Before the course starts, the teacher is responsible for informing candidates of WJEC's regulations concerning malpractice. Candidates must not take part in any unfair practice in the preparation of work for GCSE Computer Science.

Information regarding malpractice is available in our Guide to preventing, reporting and investigating malpractice.

All cases of suspected or actual malpractice must be reported immediately to WJEC (malpractice@wjec.co.uk). If candidates commit malpractice, they may be penalised or disqualified from the examinations.

In all cases of malpractice, centres are advised to consult the JCQ booklet Suspected Malpractice: Policies and Procedures.

5 TECHNICAL INFORMATION

5.1 Entries and Awards

This is a unitised qualification. Candidates are entered for each unit separately.

Assessment opportunities will be available in the summer assessment period each year, for the lifetime of the specification.

Unit 1 will be available in 2026 (and each year thereafter). Unit 2 will be available in 2027 (and each year thereafter) and the qualification will be awarded for the first time in summer 2027.

Pre-16 Candidates (i.e. candidates who are 16 or under on 31st August in the academic year in which they were entered)

The terminal rule is set at 40% of the overall qualification for Pre-16 Candidates for GCSE Computer Science.

If the assessment being re-taken contributes to the 40% terminal rule requirement, the mark for the new assessment will count regardless of previous results in that assessment.

Candidates may resit an individual unit once only. The better uniform mark score from the two attempts will be used in calculating the final overall grade subject to the terminal rule being satisfied first i.e. that candidates must complete 40% of the assessment for a qualification in the series in which they are cashing in.

If any unit has been attempted twice and a candidate wishes to enter the unit for the third time, the candidate will have to re-enter all units and the appropriate cash-in(s). This is referred to as a 'fresh start'. When retaking a qualification (fresh start), a candidate may have up to two attempts at each unit. However, no results from examination units taken prior to the fresh start can be used in aggregating the new grade(s).

If a candidate has been entered for but is absent for a unit, the absence does not count as an attempt. The candidate would, however, qualify as a resit candidate in the final resit series.

Post-16 Candidates (i.e. candidates who are 16 or over on 1st September in the academic year in which they are entered)

There is no terminal rule that applies to Post-16 Candidates for GCSE Computer Science.

There is no limit on the number of times a candidate can resit an individual unit; however, the better uniform mark score from the most two recent attempts will be used in calculating the final overall grade.

The ‘fresh start’ rule does not apply to post-16 candidates.

If a candidate has been entered for but is absent for a unit, the absence does not count as an attempt. The candidate would, however, qualify as a resit candidate in the final resit series.

The entry codes appear below:

		Entry codes	
		English medium	**Welsh medium**
Unit 1	Understanding Computer Science	3460U1	3460N1
Unit 2	Computer Programming	3460U2	3460N2
WJEC GCSE Computer Science		3460QS	3460CS

The current edition of our Entry Procedures and Coding Information gives up-to-date entry procedures.

5.2 Grading, awarding and reporting

GCSE qualifications are reported on an eight-point scale from A*-G, where A* is the highest grade. Results not attaining the minimum standard for the award will be reported as U (unclassified).

Individual unit results are reported on a uniform mark scale (UMS) with the following grade equivalences:

	MAX.	**A***	**A**	**B**	**C**	**D**	**E**	**F**	**G**
Unit 1	100	90	80	70	60	50	40	30	20
Unit 2	100	90	80	70	60	50	40	30	20
Subject Award	200	180	160	140	120	100	80	60	40

Appendix A: Opportunities for embedding elements of the Curriculum for Wales

Curriculum for Wales Strands	Unit 1	Unit 2
Cross-cutting Themes		
Local, National & International Contexts	✓	✓
Sustainability aspect of Local, National & International Contexts	✓	
Relationships and Sexuality Education	✓	
Human Rights	✓	
Diversity		
Careers and Work-Related Experiences	✓	✓
Cross-curricular skills - Literacy		
Listening	✓	✓
Reading	✓	✓
Speaking	✓	✓
Writing	✓	✓
Cross-curricular skills - Numeracy		
Developing Mathematical Proficiency	✓	✓
Understanding the number system helps us to represent and compare relationships between numbers and quantities	✓	
Learning about geometry helps us understand shape, space and position and learning about measurement helps us quantify in the real world	N/A	N/A
Learning that statistics represent data and that probability models chance help us make informed inferences and decisions	N/A	N/A

Curriculum for Wales Strands	Unit 1	Unit 2
Digital Competence		
Citizenship	✓	✓
Interacting and Collaborating	✓	✓
Producing	✓	✓
Data and Computational Thinking	✓	✓
Integral Skills		
Creativity and Innovation	✓	✓
Critical Thinking and Problem Solving	✓	✓
Planning and Organisation	✓	✓
Personal Effectiveness	✓	✓

Appendix B

Conventions followed in specification

Note 1

Where Von Neumann architecture is represented diagrammatically, the following symbols are used:

Arithmetic logic unit 

Control unit 

Register 

Note 2

Where candidates are required to apply computing-related mathematics, the following arithmetical and relational operators will be used:

Operator	Meaning	Example
>	Greater than	`A > B` will return `TRUE` if the value of A is higher than the value of B otherwise it will return `FALSE`.
<	Less than	`A < B` will return `TRUE` if the value of A is lower that the value of B otherwise it will return `FALSE`.
<=	Less than or equal to	`A <= B` will return `TRUE` if A is the same as or lower than B otherwise it will return `FALSE`.
>=	Greater than or equal to	`A >= B` will return `TRUE` if A is the same as or higher than B otherwise it will return `FALSE`.
<>	Not equal to	`A <> B` will return `TRUE` if A is not the same as B but `FALSE` if A is the same as B.
=	Equal to	`A = B` will return `TRUE` if A is equal to B otherwise it will return `FALSE`.
`AND`	Both statements must be true for the argument as a whole to be true.	`(A = 1) AND (B = 4)` will return `TRUE` if A is 1 and B is 4. It would return `FALSE` in all other situations.

`OR`	Only one of the statements needs to be true for the argument as a whole to be true.	`(A = 1) OR (B = 4)` will return `TRUE` if A is 1 or B is 4. It would only return `FALSE` if A is not 1 and B is not 4.
`NOT`	The opposite of	`NOT(A)` will return `TRUE` if A is `FALSE` and `FALSE` if A is `TRUE`.
`XOR`	The argument is false if both statements are true. The argument is false if both statements are false. Otherwise, the statement is true.	`A XOR B` would return `TRUE` if A and B are different values.
`DIV`	Integer division Finds the quotient or the 'whole number of times' a divisor can be divided into a number.	`11 DIV 2 = 5` The quotient is 5 as 2 divides into 11 a whole number of 5 times.
`MOD`	Modulo division Finds the remainder when a divisor is divided into a number.	`11 MOD 2 = 1` The remainder is 1 as 2 divides 5 times into 11 with '1 remaining'.

Note 3

Algorithms written in pseudo code will be represented using the following convention:

Construct	Example usage
Declare subroutines	`Declare CapitalLetterOfName` `End Subroutine`
Call a subroutine	`call SubroutineNeeded`
Declare and use arrays	`myarray[99]`
Literal outputs	`output "Please enter a number"`
Variable names	`myvariable`
Define variable data type	`myvariable is integer`
Data types	`integer, character, string, boolean, real`
Assignment	`set counter = 0`
Selection	`if . . . else . . . end if`

Indent at least single space after if or do or repeat etc.	`if counter = 1`
	`output counter`
	`end if`

Comments	`/** Comments for program */`

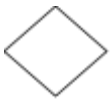
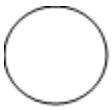
Repetition	`for i . . . next i`
	`repeat . . . until`
	`do . . . loop`
	`do . . . while`
	`while . . . repeat`

String handling	`mid(string,x,y)`
	`left(string,x)`
	`right(string,x)`
	`instring(x,stringa,stringb)`
	`len(string)`
	`val(string)`
	`int(string)`
	`trim(string)`
	`char(number)`

Logical operators `AND OR NOT XOR` will be in upper case.
Logical `TRUE` and `FALSE` will be in upper case.

Note 4

Algorithms represented using a flowchart will use the following symbols as part of the convention:

Start / Stop procedure	
Decision box	
Input / output	
Operation	
Connector	
Store / Subroutine call	
Flow of control (Arrowhead indicates direction of flow)