GCSE

WJEC GCSE in

COMPUTER SCIENCE

APPROVED BY QUALIFICATIONS WALES

GUIDANCE FOR TEACHING

Teaching from 2017

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

Contents

Introduction

It will be awarded for the first time in summer 2019, using grades A*- G.

Our GCSE Computer Science specification has three units. It is designed to allow learners to apply their knowledge of the theory of computer science and to link this seamlessly with programming and computational thinking.

The specification builds on the reputation WJEC has established with its current GCSE in Computer Science for clear, reliable assessment supported by straightforward, accessible guidance and administration. We have a proven track record of successfully providing assessment that caters for a wide range of ability and successfully motivates many learners towards further study in the subject.

Key features include:

- Opportunities for flexible teaching approaches
- Straightforward wording of questions
- Accessibility of materials
- A range of question types
- Opportunities for producing extended writing
- Opportunities to link theoretical and practical concepts
- High-quality examination and resource materials

The full set of requirements is outlined in the specification which can be accessed on the WJEC website.

In addition to this Guidance, support is provided in the following ways:

- Specimen assessment materials
- Face-to-face CPD events
- Examiners' reports on each question paper
- Free access to past question papers and mark schemes via the secure website
- Direct access to the Subject Officer
- Free online resources
- Exam Results Analysis
- Online Examination Review

Aims of the Guidance for Teaching

The principal aims of the Guidance for Teaching are to offer support to teachers in their delivery of the new WJEC GCSE in Computer Science specification and offer guidance as to the requirements of the qualification and the assessment process.

The Guidance is **not intended as a comprehensive reference**, but as support for teachers to develop stimulating and exciting courses tailored to the needs and skills of their own students in their particular institutions.

The Guidance offers assistance to teachers with support from the Principal Examiners who highlight areas of their components as an expansion of the information in the specification

Information for teachers transitioning from the 2012 WJEC GCSE Computer Science Specification

This section details topics that have changed from the 2012 WJEC GCSE Computer Science Specification. These changes fall into the following three categories:

1. **Topics that have been removed**
2. **Topics that include more detail**
3. **New topics**

Categories 1 and 2 above are not mutually exclusive; as a removed topic may have been replaced by a different topic that covers content originally defined under a different heading, but has been strengthened. For example IP and TCP (as separate protocols) and protocol stacks have been removed but in their place TCP/IP and the 5 layer model have been included.

1. Topics that have been removed

- Storage in the cloud
- The nature of data
- Why data needs to be converted into binary format
- How the computer knows if it's reading instructions or data
- Libraries
- Disk organisation
- Applications (WP/Spreadsheet/presentation/database/drawing)
- IP and TCP (as separate protocols), protocol stacks
- Necessary hardware to connect to the internet
- Common file standards associated with the internet
- Data redundancy
- MAC addresses
- Why HTML is important for standard web page creation
- How search engines work
- Interpret and dry run simple algorithms (replaced with more detailed breakdown)
- The need for accuracy in both algorithm and data
- Complete & test algorithms (replaced with more detailed breakdown)
- Translator (replaced with more detailed breakdown of compilers, assemblers, interpreter)
- Strategies to avoid errors which may occur in programming code
- Small domain specific languages
- Design and write solutions to given problems such as: write an app, a game or a tool to perform a task (replaced by far more detailed software development section)
- Debug programs (see above reference to more detailed software development section)
- Explain how a program achieves its intended result (see above reference to more detailed software development section)
- Use strategies for overcoming and detecting problems caused by mistakes in the original program.

2. Topics that include more detail

- Von Neumann architecture explicitly stated
- Explain the difference between RISC and CISC types of processors explicitly defined
- Decode now explicit in fetch-execute cycle
- Advantages and disadvantages of network topologies stated
- Wired and wireless communication explicit
- Protocols more explicit (Ethernet, Wi-Fi, HTTPS and email protocols (e.g. POP3/IMAP/SMTP)
- TCP/IP packets content explicit
- Digital storage of graphics and sound explicit
- Justification of data structures explicit
- The need to design, interpret and manipulate data structures now explicit
- Role of the operating system explicit in terms of the functions it provides, e.g. resources and interface
- Utility software now a discrete element
- Programming constructs now explicit and include counts, rogue values, sequence, selection, iteration
- Algorithms now streamlined to 'follow, make alterations to and write'
- Data types, constants and variables explicit in algorithms
- Scope of variables included explicitly in algorithms
- Self-documenting identifiers and annotation explicit in algorithms
- Mathematical operations included explicitly in algorithms
- Understanding the purpose of high and low level languages now explicit and identifying where they could be used
- IDE now specified and its role in developing/debugging programs
- The marking grids for the NEA is significantly more detailed
- Cybersecurity explicit and broken into malware types, forms of attack, methods of identifying vulnerabilities.

3. New topics

- Calculate data capacity requirements (e.g. of a file)
- Role of additional hardware (GPU, sound card etc.)
- Embedded systems
- Use XOR logical operators
- Boolean expressions, rules and identities
- Circuit switching
- TCP/IP and 5 layer model
- Calculate routing cost
- Binary addition
- Shifts
- Records
- Merge and Bubble sort
- Linear and Binary search
- Validation and authentication (security)
- Cybersecurity.

Guidance for Teaching

The following section is intended to give teachers guidance on the specification. It is designed to stimulate ideas for teaching and possible subject areas that may be explored to expand upon the content of the specification. It is not an exhaustive list and is not meant to be a substitute for sound classroom teaching.

Teachers should use this expansion in conjunction with class notes, exercises and textbooks or similar resources.

Sections have lesson ideas, where appropriate, at the end of sections. It is hoped that these will help in the delivery of the subject; however there is no requirement to use these ideas. Should you have any ideas that you feel could be added to this Guidance, please send these to resources@wjec.co.uk with 'Computer Science GCSE lesson idea' in the message line.

Unit 1

1. Hardware

Topic	Amplification	Teacher Guidance
Architecture	Describe the characteristics of CPU architecture, including Von Neumann architectures.	Arithmetic logic unit, control unit (clock), data/address/control bus within the Von Neumann architecture. Knowledge of the existence of the Harvard architecture.
	Identify and explain the role of the components of the CPU in the fetch-decode-execute cycle.	Know that instructions have to be fetched from memory, decoded within the CPU and then executed. Registers: • Program Counter (PC) – a counter that keeps track of the memory address of which instruction is to be executed next. • Memory Address Register (MAR) – the address in main memory that is currently being read or written. • Current Instruction register (CIR) – a temporary holding area for the instruction that has just been fetched from memory. Components: • Control Unit (CU) / Clock – decodes the program instruction in the CIR, selecting machine resources and a particular arithmetic operation, and coordinates activation of those resources. • Arithmetic logic unit (ALU) – performs mathematical and logical operations.

	Explain how performance is affected by the cache size, clock speed and number of cores.	Cache size • More cache memory improves the performance as it can provide instructions and data to the CPU at a much faster rate than other system memory such as RAM. • More cache memory will allow more instructions that are repeatedly used by a CPU to be stored, and therefore increase the hit rate; increasing performance as a result. Clock speed • The faster the clock speed, the faster the computer is able to run the fetch-decode-execute cycle and therefore process more instructions. • The faster the clock speed, the more power is generally required which creates greater requirements for heat dissipation and can place more strain on battery life. • Impact on temperature can be damaging. Number of cores • In a single-core CPU each instruction is processed one after the other, whereas in a dual-core CPU, two instructions may be processed at the same time. In theory, dual-core CPU should mean that the computer can process instructions twice as fast as a single-core CPU. • Performance may be affected where one core is waiting on the result of another and therefore cannot carry out any more instructions, leading to the performance being no better than a single core processor. • However multiple cores increase processor cost.

	Explain the difference between RISC and CISC types of processors.	RISC has fewer instructions than CISC To perform complex tasks; RISC CPUs must combine simple operations so RISC can be more efficient at performing simpler tasks. In order to handle complex instructions CISC CPUs are physically larger to accommodate more complex circuitry. RISC CPUs are designed to use less power and run cooler meaning that they can be used in smartphones that don't have dedicated cooling systems (like fans). RISC CPUs tend to be cheaper to mass produce. RISC CPUs run at lower clock speeds than CISC CPUs. They can perform simpler tasks more quickly than CISC, but are generally not used to carry out complex instructions.
Input / output	Describe the use and characteristics of input and output devices.	Understand and relate the use of input devices to given contexts. For example: • When typing documents, a keyboard is the preferred device. • For someone with a disability, such as paralysis, a sip/puff tube may be used. • In a noisy environment, voice input would not be useful.
Primary storage	Explain the functional characteristics of Random Access Memory (RAM), Read Only Memory (ROM), flash memory and cache memory.	Understand that all memory stores information as bits and bytes. RAM is a volatile memory type, which stores programs and data currently in use. ROM is non-volatile memory type, which stores programs and data that does not change. Flash memory can be NAND memory. Flash memory is used for the permanent storage of data. This means that data is not lost when the power is switched off. However, the data stored in flash memory can be changed. Cache memory is volatile and can be used for the temporary storage of frequently accessed data and instructions. It consists of a small number of store locations that can be accessed very quickly by the CPU; quicker than RAM. Understand and be able to give uses for these types of memory in different contexts.

Secondary storage	Describe the characteristics of contemporary secondary storage technologies including magnetic, optical and solid state.	Magnetic storage – disk/tape Optical storage – CD/DVD/Blu-ray Solid state storage – Flash / SD
	Explain the functional characteristics of contemporary secondary storage devices in terms of suitability, durability, portability and speed.	Relate secondary storage types to a situation or use. Suitability relates to overall suitability to a task and can be based on any of the below: Durability can relate to resistance to damage or longevity. Portability relates to removability and size. Speed relates to access time. For example: Magnetic storage: • Media: hard disk drives • Suitability: Backing up a home computer system • Capacity: 500 MB – 4 TB • Durability: ★ • Portability: Yes • Speed of access: ★★★ Optical storage: • Media: CD/DVD/Blu-ray • Suitability: Storing multimedia files • Capacity: 650 MB (CD), 9GB (DVD), 50 GB (Blu-ray) • Durability: ★★★ • Portability: Yes • Speed of access: ★★ Solid state: • Media: Flash memory drives • Suitability: Moving relatively small files from work to home • Capacity: 2 GB – 64 GB • Durability: ★★★★ • Portability: Yes • Speed of access: ★★★★

<table>
<tr>
<td>Storage requirements</td>
<td>Describe the relationship between data storage units, including bit, nybble, byte, kilobyte and additional prefix multipliers.</td>
<td>Bit as <code>1</code> or <code>0</code>

Nybble as 4 bit block (and usefulness in conversion from hex-binary) e.g. $1101_2 \rightarrow$ C

Byte as 8 bits 10110110_2

Understand that prefix multipliers work from bytes as 2^3 increments.

<table>
<tr><th></th><th>Symbol</th><th>Value</th></tr>
<tr><td>Byte</td><td>B</td><td>8 bits</td></tr>
<tr><td>Kilobyte</td><td>KB</td><td>1024 bytes</td></tr>
<tr><td>Megabyte</td><td>MB</td><td>1024 Kb</td></tr>
<tr><td>Gigabyte</td><td>GB</td><td>1024 MB</td></tr>
<tr><td>Terabyte</td><td>TB</td><td>1024 GB</td></tr>
<tr><td>Petabyte</td><td>PB</td><td>1024 TB</td></tr>
<tr><td>Exabyte</td><td>EB</td><td>1024 PB</td></tr>
<tr><td>Zettabyte</td><td>ZB</td><td>1024 EB</td></tr>
<tr><td>Yottabyte</td><td>YB</td><td>1024 ZB</td></tr>
</table>
</td>
</tr>
<tr>
<td></td>
<td>Describe data capacity and calculate data capacity requirements.</td>
<td>Understand data capacity in terms of the amount of data (measured in bytes) that can be stored by a given medium.</td>
</tr>
<tr>
<td>Other hardware components</td>
<td>Describe the characteristics and role of additional hardware, including GPU, sound cards and motherboards.</td>
<td>Understand that additional hardware is required by computers to execute tasks.

For example, the GPU is a specialised electronic circuit designed to rapidly manipulate and alter memory. GPUs efficiently manipulate computer graphics and carry out image processing.</td>
</tr>
<tr>
<td>Embedded systems</td>
<td>Describe the use and give examples of embedded systems.</td>
<td>Understand that embedded systems exist in many devices in the home and outside and that these are small, dedicated computers. Embedded systems tend to be stored on ROM and are permanent (except in some cases where firmware is held on Flash memory and can be changed).

Examples could include washing machines (the embedded system contains the washing programs), remote garage door entry systems, and engine sensors.</td>
</tr>
</table>

Lesson ideas:

Use an old PC – This will take place over a number of lessons

Architecture: Sometimes the scale of the CPU and internal components can be difficult to grasp. If you can obtain a PC which is not currently in use (most schools have to retain these until registered as discarded) it can be interesting to pull one apart. The key points here are to show that the CPU is a small integrated system and that RAM is separate. (Pull off the fan and heat sink, scrape off the thermal paste and pop the CPU out – then point out that the CPU is the tiny chip in the middle of the assembly.

Discuss that this tiny chip contains the CU, the ALU and the registers. Reinforce that the chip contains its own memory (registers and cache) but that the RAM is separate from the chip and this stores active programs while they are being run. This is an opportunity to explain the concepts of the CU, ALU and the registers/RAM.

Bits and Bytes: You can explain that RAM holds data stored as bits and bytes. Explain the concept of bits as 1 and 0 (on and off) as this is all essentially a computer can 'understand'. Explain that different combinations of 1 and 0 can represent an infinite range of data and that data is stored as bytes, kilobytes, megabytes etc.

Secondary Storage: There are opportunities here to discuss secondary storage drives whilst dismantling the PC and to see how these connect to the motherboard.

Discuss the suitability of different storage types in terms of capacity, portability, durability and speed of access.

It could be an idea to have a 'storage race' to discuss speed. Compare the transfer time of a large (>2GB) file onto magnetic HDD vs Optical vs Flash memory. The properties of the storage devices can be looked at here to look at capacities (e.g. the HDD will probably have a high GB or TB storage range whereas a DVD would only have 4.7GB, Flash memory would be variable but normally lie between the two).

CISC and RISC: While discussing the CPU, point out that there are different types of CPUs, including RISC and CISC. PC, invariably use the CISC architecture, whereas mobile phones tend to use a RISC architecture. Explain the difference between RISC and CISC instruction sets.

Clock speed: You may also want to discuss clock speeds and that it is possible to overclock. This allows you to explain clock speed, and issues with overclocking (relate back to the heat sink and fan that you've already taken off the CPU).

Number of cores: Ask five students to volunteer to aid you in demonstrating the advantages of parallel processing. Explain to the class that a task such as sorting a deck of playing cards into order (A–K → Clubs, Diamonds, Hearts and Spades), can be completed much faster by four people (Quad-core processor) than by a single person (Single-core processor).

Give one deck to the student representing a single-core processor and give a quarter of the other deck to each student in the group of four representing a quad-core processor.

Both groups have to sort the deck as fast as possible

Explain to the class that large problems can be divided into smaller ones, which are then solved concurrently and faster.

Additional hardware components: The graphics card, sound card (may be integrated) and other can be shown in this exercise.

Embedded Systems: Point out the BIOS on the motherboard, explain that this is an example of ROM and it stores simple start-up sequences. Go on to explain that ROMs such as this can hold any program. Relate this to ROMs in washing machines etc. Explain that they may not always be ROMs and are sometimes NAND memory that can contain updateable firmware (e.g. car ECUs) that can be 'flashed' in order to update it.

2. Logical operations

<table>
<tr><th>Topic</th><th>Amplification</th><th colspan="3">Teacher Guidance</th></tr>
<tr><td>Logical operators</td><td>Use AND, OR, NOT and XOR logical operators, combinations of these, and their application in appropriate truth tables to solve problems.</td><td colspan="3">Understand the basic principles of the 4 logical operations, for example:
1 AND 0 = 0
1 OR 0 = 1
NOT 1 = 0 (or NOT 0 =1)
1 XOR 0 = 1
Use these in combination for example:
(1 AND 0) OR (1 AND 1) = 1
NOT (1 XOR 0) = 0</td></tr>
<tr><td rowspan="11">Boolean logic</td><td rowspan="11">Simplify Boolean expressions using Boolean identities and rules.</td><td colspan="3">Use the following Boolean identities and rules:</td></tr>
<tr><td></td><td>AND Form</td><td>OR Form</td></tr>
<tr><td>Commutative Law</td><td>A. B = B . A</td><td>A + B = B + A</td></tr>
<tr><td>Associate Law</td><td>(A. B) . C = A . (B . C)</td><td>(A + B) + C = A + (B + C)</td></tr>
<tr><td>Distributive Law</td><td>(A+B)+ C = (A+C).(B+C)</td><td>(A + B) . C = (A . C) + (B . C)</td></tr>
<tr><td>Identity Law</td><td>A . 1 = A</td><td>A + 0 = A</td></tr>
<tr><td>Zero and One Law</td><td>A . 0 = 0</td><td>A + 1 = 1</td></tr>
<tr><td>Inverse Law</td><td>A . A′ = 0</td><td>A + A′ = 1</td></tr>
<tr><td>Idempotent Law</td><td>A . A = A</td><td>A + A = A</td></tr>
<tr><td>Absorption Law</td><td>A(A+B) = A</td><td>A + A. B = A
A + A′ B = A+B</td></tr>
<tr><td>Double Complement Law</td><td colspan="2">$\bar{\bar{x}} = x$</td></tr>
</table>

Lesson ideas:

Logical operations - Operation chains:

First point out the difference between AND and OR in terms of a meal

'Do you want a sandwich **and** a drink?' differs from 'Do you want a sandwich **or** a drink?'

Once you have established that AND and OR are different, explain that:

1 AND 0 = 0, 0 AND 1 = 0, 1 AND 1 = 1, 0 AND 0 = 0 (explain that both must be 1 for the result to be 1)

1 OR 0 = 1, 0 OR 1 = 1, 1 OR 1 = 1, 0 OR 0 = 0 (explain that if there is one or more 1, then the result will be 1)

Next explain that XOR is a different operation where 'one or the other but never both' applies and that:

1 XOR 0 = 1, 0 XOR 1 = 1, 1 XOR 1 = 0, 0 XOR 0 = 0 (explain only one 1 will give a result of 1)

Finally explain that NOT 'reverses' a result

NOT 1 = 0, NOT 0 = 1

Now try an operation chain with the class. Give a sequence of logical operations, and call them out to the class/ write them on the board/ show on a presentation. For example start with this chain, do not give the answers A–D:

1 OR 0 = **AnswerA** (1)

AnswerA AND 0 = **AnswerB** (0)

AnswerB XOR 0 = **AnswerC** (1)

NOT **AnswerC** = **AnswerD** (0)

Finally, end with a flourish (this will draw on BODMAS, which you may have to teach if they haven't been taught it in mathematics yet).

NOT (**AnswerD** XOR 1) = **AnswerE** (1)

You can now ask the students about the result they have (this chain should have 1 as its final result), and work through the problem. Some will have got it wrong, but working through will help their understanding. Repeat with more chains in class, or as independent learning tasks.

Truth tables will be the next stage once students have an appreciation of simple operations. Explain that truth tables are shorthand for the work they have already done.

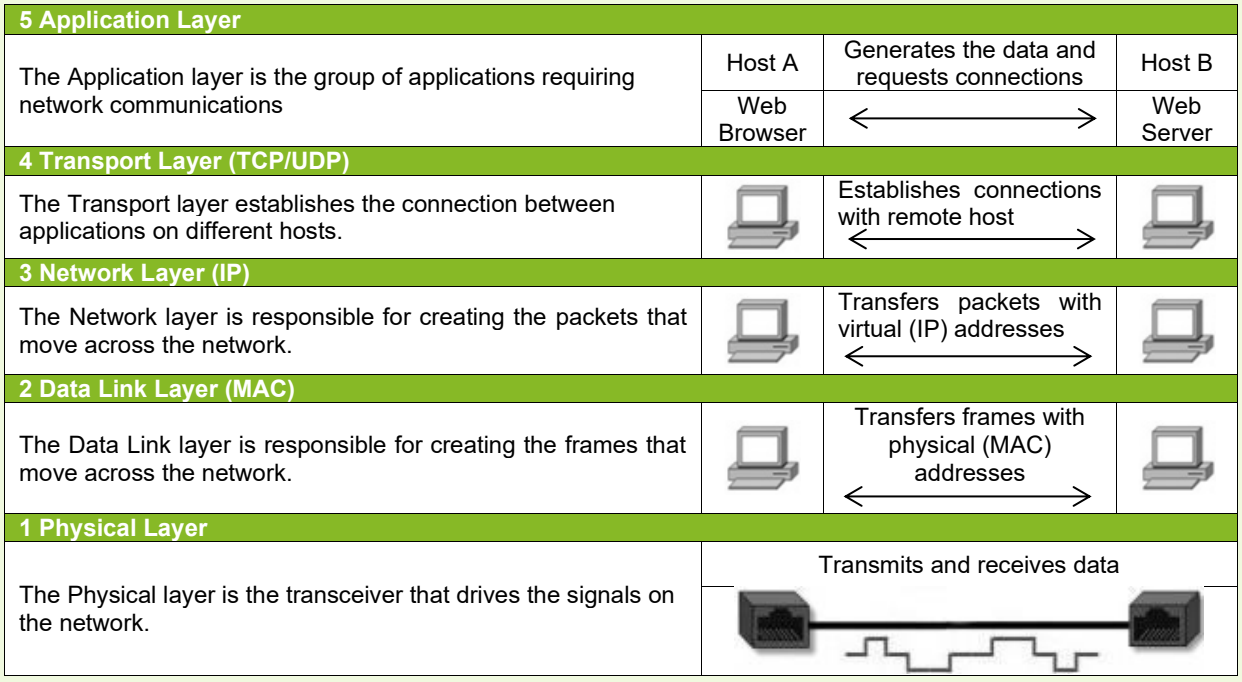
3. Communication

Topic	Amplification	Teacher Guidance
Networks	Explain the characteristics of networks and the importance of different network types, including LAN and WAN.	Explain the difference between a wide area (WAN) and local area networks (LAN) in terms of distribution. Local area being sited on a single site and wide area being sited on multiple sites. Specialist hardware is used to construct networks, such as: Switches • A switch analyses each packet of data and sends it to the computer it was intended for. Hubs • A hub copies all packets of data to all devices on the network. Routers • A router stores the addresses of computers on the network and transfers data between devices. Gateways • A gateway joins together two networks that use different base protocols, e.g. links a LAN to WAN. Bridge • A bridge joins together two networks that use the same base protocols, e.g. links LAN to LAN. Other types of network are Personal Area Networks (PAN) (very small area), Metropolitan area networks (MAN) (within a metropolitan district) and Virtual Private Networks (VPN) (allows people to log into a network remotely and access its resources, which is an implementation of a WAN).

	Describe the importance of common network topologies, including ring, star, bus and mesh, and their advantages and disadvantages.	Understand the configuration and describe the importance of: Ring – where each node connects to exactly two other nodes, providing a single pathway for signals through each node. High transfer speeds but can fail if one node fails. Bus - in which nodes are directly connected to a common linear cable (or bus). Cheap and easy to set up but can be slow under heavy traffic (due to collisions) and a break in the main bus will break the network. Star – Where one or more central switch, hub or computer acts as a central conduit to transmit messages. Very reliable and high data transfer speeds are possible (fewer collisions) and easy to identify faults. Can be expensive to set up as switches and cabling expensive, if main switch fails, the network fails and bottlenecking can occur if too much data is passing through the central switch. Mesh - each node relays data for the network. All mesh nodes cooperate in the distribution of data in the network. This is very reliable and a network can 'self-heal' by reconfiguring itself around broken paths. A network is complex, expensive and difficult to set up. A large part of the network may be redundant.
	Explain the importance of connectivity, both wired and wireless.	Connectivity is important for the transfer of data. Without connectivity, data could not be easily exchanged between computers without the use of removable storage. Wired connections use a wired connection protocol e.g. Cat6 cabling using Ethernet. Wireless does not use cabling but requires both the transmitting and receiving machine to have wireless network adapter cards and normally additional wireless routing equipment is required (not with P2P). Wireless is generally slower than wired in terms of bitrate although the advantage of greater freedom can offset this.

	Explain and give advantages and disadvantages of circuit switching and packet switching.	Understand that circuit switching was a globally accepted standard until superseded by packet switching. Circuit switching requires a series of connections to be made to form a single route and that all data traverses the same route. This is susceptible to interception and failure as failure of any one connection results in failure of the entire route. Packet switching differs in that the data is broken into packets that all traverse different routes. The data is reassembled once it arrives at the destination. Packet switching is less susceptible to interception and is more robust because if a route fails then the packet can use an alternate route.
	Explain the importance and the use of a range of contemporary network protocols, including Ethernet, Wi-Fi, TCP/IP, HTTP, HTTPS, FTP and email protocols.	Network protocols are vital to allow computers on networks to communicate. Without shared common protocols, computers would not be able to communicate. There are many types of protocol: Ethernet – wired (cable connection) protocol. Wi-Fi – wireless, two common standards are Bluetooth and 801.11. TCP/IP– transmission control protocol/ internet protocol – is the basic communication language or protocol of the Internet. HTTP – hypertext transfer protocol – allows webpages to be shared across different computers and browsers. HTTPS – A secure variant of HTTP – it works together with another protocol, Secure Sockets Layer (SSL), to transport data securely. POP3 – post office protocol 3 – is a protocol for receiving email, in which email is received and stored by an email server with a client downloading messages when ready. SMTP – simple mail transfer protocol -mail servers use SMTP to send and receive mail messages, mail applications typically use SMTP only for sending messages to a mail server. IMAP – internet message access protocol - transfers emails between computer systems via the internet. The IMAP protocol however is generally used for email retrieval and storage as an alternative to POP.

	Describe the typical contents of a TCP/IP packet.	The typical contents of a TCP/IP packet are: • The source address • The destination address • Information which enables the data to be reassembled into its original form • Other tracking information • The data itself • A checksum that checks that the data has not been corrupted
	Explain the importance of layers and the TCP/IP 5-layer model.	The TCP/IP protocol, on which the Internet is built, is not a single protocol but rather an entire suite of related protocols. The 5 layers of the TCP/IP model are: • Application • Transport • Network • Data link • Physical



	Describe methods of routing traffic on a network and calculate routing costs.	Understand that during routing, networks will search for the shortest path and the fastest nodes to transfer data. Together the path between nodes and the speed of the nodes are assessed by the device transmitting data. Computers will look for the route with the lowest *cost* (that is the shortest path and fastest nodes) and transmit data via this lowest cost route.
Internet	Explain how Domain Name System (DNS) servers and Internet Protocol (IP) addresses work.	Understand how DNS servers are used to resolve web addresses into IP addresses. Understand that IP addresses are difficult to remember compared to web addresses and that this is why DNS is vital to the ease of use of web addresses.

4. Organisation and structure of data

Topic	Amplification	Teacher Guidance
Representation of numbers	Use and convert between denary, binary (up to 16 bits) and hexadecimal counting systems.	Convert from: Denary to binary Denary to hexadecimal Binary to denary Binary to hexadecimal Hexadecimal to denary Hexadecimal to binary Direct conversion or using an intermediate base are both acceptable
	Explain the use of hexadecimal notation as shorthand for binary numbers.	Understand that a binary number is far easier to use as the shorter hexadecimal notation. e.g. $110110101011110_2 = 6D5E_{16}$
	Use arithmetic shift functions and explain their effect.	Understand the effect of shifts both left and right. <u>Left shift</u> e.g. for the number 001100_2 (12_{10}) a shift one place to the left would give $011000_2 = 24_{10}$ Which is equivalent to multiplying by 2 a shift two places to the left would give $110000_2 = 48_{10}$ Which is equivalent to multiplying by 4 <u>Right shift</u> e.g. for the number 001100_2 (12_{10}) a shift one place to the left would give $000110_2 = 6_{10}$ Which is equivalent to dividing by 2 a shift two places to the left would give $000011_2 = 3_{10}$ Which is equivalent to dividing by 4

	Apply binary addition techniques.	Use the add/divide/remainder method to add binary numbers.
	Explain the concept of overflow.	Understand that if the result of an addition or shift process results in a number that is too large to fit in the space available then an overflow has occurred. For example if we tried to store the addition of the following two 8 bit numbers in an 8 bit register. 11011011 11111011 + 111010110 The answer is too large to fit into the register (it is 9 bits in length). Where this happens, an overflow is said to have occurred.
Representation of graphics and sound	Explain the digital storage of graphics.	Understand that both raster and vector graphics can be stored on a computer. Understand that raster graphics are dot matrix data structures representing a grid of pixels and cannot scale up without loss of apparent quality. They tend to be large in terms of the memory required to store them. A bitmap image is a type of raster image and is composed of many tiny parts, called pixels, which are often many different colours. It is possible to edit each individual pixel. Vector graphics use geometrically primitive objects (geometric primitives) such as points, lines, curves, and shapes or polygons which are based on mathematical expressions to represent images. Vector graphics can be scaled up without loss of apparent quality. They are smaller than bitmap graphics in terms of the memory required to store them.
	Explain the digital storage and sampling of sound.	That sound is stored as a digital representation. The digital representation is achieved by sampling (signal processing). The sample quality can be affected by the sample rate and sample frequency. The higher the sample rate and frequency, the larger the resultant sample.

	Describe the use of metadata in files.	Understand and be able to give examples of the types of metadata that are stored with files e.g. location, date etc. within an image or in a sound file, the file or artist, recording date, song title etc.
Storage of characters	Describe how characters are stored as a binary number.	Understand that characters are represented by binary numbers e.g. in ASCII **01100001** represents **a** **01000001** represents **A** (There is no need to remember the number that represents a character).
	Describe standardised character sets, including Unicode and American Standard Code for Information Interchange (ASCII).	Understand that standardised character sets allow for data interchange between different programs. Understand that ASCII and Unicode are two such standards. Unicode can represent more characters than ASCII, but is more memory-hungry.
Data types	Describe the concept of data types, including integer, Boolean, real, character and string.	Integer (whole number) 7, 0, 15, -5 Boolean (true/false) Real (number with, or without, fraction) 7.2, 8.9, -6.8, 12.0 Character (single character) a, @, #, 8, Q String (one or more character) Hello, abc, b, Y
Data structures	Describe, design, interpret and manipulate data structures including records, one-dimensional and two-dimensional arrays.	Describe relates to being able to describe and identify a data structure (for instance drawing a representation of a one-dimensional array or a two-dimensional array). Design relates to being able to design an array or record that would be suitable for a particular purpose (e.g. to design an array suitable for storing sales data over a time period). Interpret relates to being able to utilise a data structure to select data or to convert a graphical representation of an array to a programmed form or vice versa. Manipulate relates to adding, deleting and editing data within an array or record.

	Select, identify and justify appropriate data structures for given situations.	This relates to selecting a correct data structure for a given situation (e.g. to design an array suitable for storing sales data over a time period) or to identify the most appropriate data structure from a selection of data structures, in each case being able to justify why the data type is suitable.
File design	Design files and records appropriate for a particular application.	As above, but in relation to files and records.
Data validation and verification	Explain and use appropriate techniques for data validation and verification.	Validation: Range check, type check, format check, presence check, lookup table, check digit. Verification: Double keying, proofreading.
	Design algorithms and programming routines that validate and verify data.	Algorithms relating to the above list.

Explaining bases

'Back to basics' – Many students will have forgotten that they learned to count in base 10, and will simply accept 'that's the way it is'.

You may want to start with the line:

'There are 10 types of people in the world, those who understand binary and those who don't' – leave it on the screen/board during the lesson. Tell the students that this is a fact, and they will hopefully be intrigued, or point out that it makes no sense.

Remind students that in primary school they will have written 'Hundreds, Tens and Units' above numbers that they were going to use in maths problems. Explain that by using this system, and 10 numbers (0-9) they had the building blocks to construct any number.

Now ask them what if they'd only had two numbers, 1 and 0. Could they then represent any number? Ask for the largest number that they could represent using just two digits. They will say '11' (unless they already understand binary).

Go on to explain that binary is a different base system. And rather than columns with 'thousands, hundreds, tens, units' the columns are divided …128, 64, 32, 16, 8, 4, 2, 1, and that with this system, any number can be represented.

Explain that binary as a system produces very long numbers that are difficult to handle and that although denary can be used as shorthand, a better system is to use base 16 (hexadecimal). Explain that we need 0-9, and some extra digits A-F, and that columns are in the order …256, 16, 1

Refer back to the *'There are 10 types of people in the world, those who understand binary and those who don't'*

5. System Software

Topic	Amplification	Teacher Guidance
Managing resources	Describe the purpose and functionality of the operating system in managing resources, including peripherals, processes, memory and backing store.	The O/S manages resources, such as: • Manages peripherals such as input and output devices • Manages printing using spooling • Manages backing store • File compression • Disk de-fragmentation • Manages memory (RAM) • Manages processes • Manages security
Providing an interface	Describe the purpose and functionality of the operating system in providing a user interface.	The O/S provides an interface by: • Provides a graphical user interface (windows, icons, menus, pointers) • Provide a command line interface • Allowing users to copy/delete/move/sort/search a file or folders • Access system settings such as hardware. • Allows users to have more than one window open • Provides user with errors/help messages. • Allows customisation of interface, e.g. change desktop background/layout • Allows user to switch between tasks (programs/windows)
Utility software	Explain the purpose and functionality of a range of utility software.	Explain the purpose and functionality of utility software, such as: • Virus scanning • Firewall • Defragmentation • Compression • System monitoring • Task management • Disk scanning and repair • System backup

6. Principles of Programming

Topic	Amplification	Teacher Guidance
Levels of computer language	Describe the characteristics and purpose of high-level and low-level languages.	Low level programming languages do not resemble natural languages, such as English or Welsh. They are made up entirely of bit patterns (instructions or data) that can be executed directly by the CPU. A high level language is a programming language that allows code to be written which is similar to a natural human language, such as English. Some programmers prefer to use high level programming languages as they are easier to understand, learn and program as commands are more English-like and identifiers can be long and meaningful. High level programming languages also allow the use of powerful commands that perform quite complex tasks such as MsgBox in Visual Basic or the SORT clause in COBOL.
	Identify and describe situations that require the use of a high-level or a low-level language.	Low level language are useful when speed of execution is critical or when writing software which interfaces directly with the hardware, e.g. device drivers. High level languages are used when the execution speed is not critical, e.g. in common productivity applications, such as a word processor.

7. Software Engineering

Topic	Amplification	Teacher Guidance
Software tools	Explain the role of Integrated Development Environment (IDE) tools in developing and debugging programs.	Understand the ways in which the IDE assists in the development and debugging of programs including: • Editor • Compiler • Interpreter • Linker • Loader • Debugger • Trace • Break point • Variable watch • Memory inspector • Error diagnostics

8. Program Construction

Topic	Amplification	Teacher Guidance
Compilers, interpreters and assemblers	Describe the purpose and give examples of the use of compilers, interpreters and assemblers.	An assembler is a program, which converts the low level assembly programming language into machine code. An interpreter is a program, which converts code one line at a time, into machine code and executes it. A compiler is a program that converts high level programs into machine code for execution at a later time (the entire program is converted, not one line at a time as with an interpreter).
	Explain the principal stages involved in the compilation process: lexical analysis, symbol table construction, syntax analysis, semantic analysis, code generation and optimisation.	Lexical analysis: • Comments and unneeded spaces are removed. • Keywords, constants and identifiers are replaced by 'tokens'. Symbol table construction: • A symbol table is created which holds the addresses of variables, labels and subroutines. Syntax analysis: • Tokens are checked to see if they match the spelling and grammar expected, using standard language definitions. This is done by parsing each token to determine if it uses the correct syntax for the programming language • If syntax errors are found, error messages are produced. Semantic analysis: • Variables are checked to ensure that they have been properly declared and used. • Variables are checked to ensure that they are of the correct data type, e.g. real values are not being assigned to integers. • Operations are checked to ensure that they are legal for the type of

		variable being used, e.g. you would not try to store the result of a division operation as an integer. Code generation: • Machine code is generated. Code optimisation: • Code optimisation may be employed to make it more efficient/faster/less resource intense.
	Describe and give examples of programing errors.	Programming errors, such as: • Syntax • Runtime/execution • Logical • Linking • Rounding • Truncation.

9. Security and data management

Topic	Amplification	Teacher Guidance
Data security	Describe the dangers that can arise from the use of computers to store personal data.	An awareness of the risks to data held on personal computers. Risks of hacking, loss to viruses, technical breakdown, interception, physical theft and data theft from discarded components.
	Describe methods that protect the security of data including access levels, suitable passwords for access and encryption techniques.	Methods such as: • Access levels permitting user access to designated functions/areas • Password design ('strong' vs 'weak' passwords) • Encryption techniques (e.g. XOR encryption.
Data management	Explain the need for file backups and generations of files.	Understand that backups protect data following primary data loss. Generations of files, e.g. the grandfather-father-son regime, allows data to be restored to a previous version following catastrophic data loss.
	Explain the need for archiving files.	Archiving is the process of storing data which is no longer in current or frequent use. It is held for security, legal or historical reasons.
Compression	Explain how lossy and lossless data compression algorithms are used.	Understand that compression reduces file size in memory terms. Lossy compression results in reduction of data quality following compression. Lossless compression results in no loss of data quality following compression.
	Calculate compression ratios.	Calculate compression ratios based on before and after file sizes (e.g. a 10MB file compressed to a 2 MB file will have a file compression ratio of 5:1). Using ratios, calculate the size of files following compression. Using data (e.g. size of destination storage medium), calculate the compression ratio required to store data.

Network security	Recognise the importance of network security and describe the dangers that can arise from the use of networks.	Knowledge of network security, including: • Antivirus software • Firewalls • Two-factor authentication • Access levels • Passwords Description of the dangers that can arise from the use of networks, such as: • Risks of hacking • Risks of viruses • Technical breakdown • Interception.
	Explain the purpose and typical contents of an acceptable use policy and disaster recovery policy.	Contents and purpose of model acceptable use policies and disaster recovery policies.
Cybersecurity	Describe the characteristics and explain the methods of protection against malware, including viruses, worms and key loggers.	Malware • Short for malicious software, malware is a broad-spectrum term used to describe software used to disrupt computer operation. Viruses • A virus is a computer program that is able to copy itself onto other programs often with the intention of maliciously damaging data. A virus is transmitted by 'piggybacking' on another program known as a 'vector'. Worm • Is similar to a virus but is a standalone program that replicates itself in order to spread to other computers. It does not need a vector. Key loggers • Are covert programs that capture keyboard (or other input device) input and transmit this data to a third party or hold the data for collection. Protection can be from: • firewalls • antivirus programs • patching out-dated software • security tools • personnel.

	Describe the different forms of attack based on technical weaknesses and/or user behaviour.	Technical weakness: • Infection by any of the programs above. • SQL injection • DoS attack. • Password-based attack. • IP address spoofing. User behaviour • Social engineering. • Phishing.
	Describe methods of identifying vulnerabilities.	Footprinting • Interrogating resources on the Internet for information about systems, looking to discover what a potential attacker can also discover without an organisation's knowledge (can remove 'enticements' or 'low hanging fruit' by this method). Penetration testing • Attempting to penetrate a system's security layers in order to demonstrate security risks.
	Explain different ways of protecting software systems during design, creation, testing and use.	Understand that software systems can be insecure and that following certain practices during design, creation and testing can mitigate these insecurities. This is commonly called 'secure by design'. Malicious practices are taken for granted and it is assumed that the system will have invalid data entered or will be the subject of a hacking attempt and therefore these issues are taken into account when creating a new system. Common examples include: • Buffer overflows • Too many permissions • Scripting restrictions • Accepting parameter without validation
	Describe the role of internet cookies.	A cookie is the term given to describe a small piece of code that is given to a Web browser by a Web server. The main purpose of a cookie is to identify users and prepare customized Web pages or to save site login information. Cookies can be seen as a security issue as they hold personal information and this can be used or sold and tracking cookies can hold information on the websites visited by users.

10. Ethical, legal and environmental impacts of digital technology on wider society

Topic	Amplification	Teacher Guidance
Ethical	Describe the ethical impacts of digital technology, including issues of privacy and cybersecurity.	Digital divide. Working conditions. Environmental impact (links to environmental issues). The implications of privacy of private data. Black hat/white hat/grey hat hacking. The growing importance of cybersecurity in light of increasing frequency of attacks.
	Explain the importance of conforming to professional standards, including formal and informal codes of ethical behaviour.	A formal code would comprise an acceptable use policy or internet access policy. Any policy that is formally written and signed up to is considered a formal code. An informal code comprises expectations, customs, habits, personal integrity.
Legislation	Explain how relevant current legislation impacts on security, privacy, data protection and freedom of information.	Current legislation including: Data protection Act. Computer misuse Act. Copyright Act. Regulation of Investigatory Powers Act. Freedom of Information Act.
Environmental issues	Describe the environmental impacts of digital technology on wider society.	Technotrash/Ewaste. Increased energy use in the home. Global assembly lines. Increased mining for rare earth elements. Huge energy consumption by server farms (servers and more significantly, cooling).

Unit 2

1. Problem solving

Topic	Amplification	Teacher Guidance
Problem solving	Use a systematic approach to problem solving including the use of decomposition and abstraction.	Decomposition is the process of breaking a complex problem into smaller component parts. Abstraction is the process of removing unnecessary detail and simplifying. Generally, abstraction precedes decomposition.
	Use abstraction effectively to model selected aspects of the external world in an algorithm or program.	Using abstraction to remove unnecessary detail from a real-world situation and to model the simplified result in an algorithm or program.
	Use abstraction effectively to appropriately structure programs into modular parts with clear, well-documented interfaces.	Using abstraction to produce a well-structured (as simple and elegant as possible) program. Modular programs should be built of self-contained subroutines. Interfaces should be clear, unambiguous and, where documented, documentation should be as unambiguous as possible.

2. Algorithms and programming constructs

Topic	Amplification	Teacher Guidance
Algorithms	Use common methods of defining algorithms, including pseudo-code and flowcharts.	See Appendix C of specification for detail on pseudo-code conventions and flowchart conventions used.
Programming constructs	Identify, explain and use subroutines in algorithms and programs.	Identify subroutines in algorithms and explain their function. Understand that subroutines are a sequence of instructions that perform a specific task. Use subroutines to solve given problems.
	Identify, explain and use sequence, selection and iteration in algorithms and programs.	Identify sequence, selection and iteration in algorithms and explain their function. Sequence: an action, or event, leads to the next ordered action in a predetermined order. Selection: In selection, a question is asked, and depending on the answer, the program takes one of two courses of action. Iteration: An iteration is a single pass through a set of instructions. Most programs contain loops of instructions that are executed over and over again. The computer repeatedly executes the loop, iterating through the loop.
	Identify, explain and use counts and rogue values in algorithms and programs.	Iterations within a loop must eventually be terminated. Counts and rogue values can be used to terminate iteration loops. Counts count the number of times a process has occurred. Rogue values are values that fall outside the range or type of the 'normal' data that will be processed by a program. In both cases a loop can be terminated if a count reaches a pre-set value or if the loop encounters a rogue value.

	Identify and explain constructs in object orientated programs.	Identify and explain the following in a given program: • superclass • class • object • property • method • comment
	Follow and make alterations to algorithms and programs that solve problems using: sequence, selection and iteration input, processing and output.	Follow the flow of and dry run algorithms, making changes to the algorithms that will affect output or repair a 'broken' algorithm. Algorithms can be of sequence, selection and iteration types and can contain input, processing and output of values.
	Write algorithms and programs that solve problems using: sequence, selection and iteration input, processing and output.	Write algorithms to solve given problems. Algorithms can be of sequence, selection and iteration types and can contain input, processing and output of values.
Variables	Identify, explain and use local and global variables in algorithms and programs.	Identify global and local variables in algorithms and be able to use these appropriately. Understand that local variables only exist for the lifetime of a subroutine and that the data in those variables is only available to the subroutine in which it is declared. Understand that global variables exist throughout an entire program and that the data in those variables is available to all routines within the entire program.
Identifiers	Explain why the use of self-documenting identifiers and annotation are important in programs. Give examples of self-documenting identifiers and annotation.	Understand that programs should be, wherever possible, named sensibly so that a third party can understand their purpose e.g. a variable containing a first name should be defined as `firstname`

String handling	Identify, explain and use routines for string handling in algorithms and programs.	String passing Concatenation String comparison Substitution Trimming Measuring length
Mathematical operations	Identify, explain and apply computing-related mathematical operations in algorithms and programs.	See Appendix C of specification for detail of computing-related mathematical operation used. Use these within algorithms and programs to process data.
Logical operations	Identify, use and explain the logical operators AND, OR, NOT and XOR in algorithms and programs.	See Appendix C of specification for detail of logical operations used. Use these within algorithms and programs to process data.
Sorting	Describe the characteristics of merge sort and bubble sort algorithms.	Merge sort as a "divide and conquer" algorithm where: An unsorted list is divided into n sublists, each containing 1 element. The sublists are repeatedly merged to produce new sorted sublists until there is only 1 sorted sublist remaining. Bubble sort as a simple sorting algorithm where: The algorithm that repeatedly steps through the list to be sorted. The algorithm compares each pair of adjacent items and swaps them if they are in the wrong order. The passes through the list are repeated until no swaps are needed

Searching	Explain and use linear and binary search algorithms.	Use linear searches and understand that this is a simple search process where a list is searched until the required value is found. Use binary search algorithms and understand binary search as a "divide and conquer" algorithm where: The middle value in a list is inspected to see if it matches the search value. If the middle value is greater than the search value, the upper half of the list is discarded. If it is less than the search value, the lower half is discarded. This process is repeated, with the list halving in size each time until the search value is found.
Testing and evaluation	Explain how an algorithm or program works and evaluate its fitness for purpose in meeting requirements.	Explain how an algorithm processes data to produce an outcome. Compare different algorithms (e.g. is a nested IF statement longer than a nested loop, if so which has been programmed more efficiently?).
	Evaluate the efficiency of an algorithm or program using logical reasoning and test data	Dry run algorithms using test data and explain the outputs.

Lesson ideas:

Binary Search vs Linear search – Ask two groups of students to guess the number you're thinking of (which is between 1 and 1000). One group should use the binary search methodology, the others linear search. Who guesses your number first? (For added fun, do this a number of times with different numbers, then pick 1 as your number to prove that a binary search is not *always* quicker).

3. Programming languages

Topic	Amplification	Teacher Guidance
Markup languages	Design, write, test and refine HTML pages using the following tags: • HTML <html> • Head <head> • Title <title> • Body <body> • Headings <h1> - <h6> • Paragraph <p> • Italic <i> • Bold <b> • Centre align • Anchor <a href="URL"> • Unordered List • List Item • Blockquote <blockquote> • Horizontal Rule <hr> • Image <img> and their corresponding closures.	Understand that tags have different effects on text and be able to identify tags that have these effects. Use the tags to mark up a document to specific requirements.

Object oriented languages	Design, write, test and refine Java programs within the Greenfoot environment, using the following skills: • Create new and extend existing classes • Create new and edit existing objects • Create new and edit existing worlds • Write and invoke methods • Change existing methods • Create new and edit existing properties (including public, private, static, etc.) • Add and remove objects from worlds • Use actors • Move objects around a world • Keyboard input • Add and play sounds • Implement and use parameter passing (by value and by reference) • Access one object from another • Implement object collision detection • Implement random number generation • Use the concept of inheritance and encapsulation.	For example, candidates could be required to: • Create a new Greenfoot world • Populate a given world with one or more objects • Edit the objects so that they turn and move randomly • Edit the program code to make the objects move in the direction of the arrow keys when pressed • Edit the objects to detect object collision and remove these objects from the world • Add a sound which will play when objects requires it to • Add objects which can receive messages from other objects • Edit objects so that they can change the image/value displayed on the Greenfoot world • Save completed worlds as file names stated.
Assembly language	Design, write, test and refine simple assembly programs using the following mnemonics: • Input INP • Output OUT • Store STA • Load LDA • Add ADD • Subtract SUB • Branch BRA • End/Stop/Halt HLT • Data definition DAT	For example, candidates could be required to design, write, test and refine simple assembly programs that: • loads register R with the contents of address X • loads register S with the contents of address Y and adds the two numbers together.

4. Data structures and data type

Topic	Amplification	Teacher Guidance
Implementing data structures	Use one-dimensional and two-dimensional arrays, files and records.	Use, as appropriate, one dimensional arrays in algorithms and programs to input, store, process and output data. Use, as appropriate, two dimensional arrays in algorithms and programs to input, store, process and output data. Use, as appropriate, records in algorithms and programs to input, store, process and output data.
Implementing data types	Use a variety of data types, including integer, Boolean, real, character and string.	Use as appropriate data types to hold data in variables/arrays/records in algorithms and programs. Integer (whole number) 7, 0, 15, -5 Boolean (True/False) Real (number with, or without, fraction) 7.2, 8.9, -6.8, 12.0 Character (single character) a, @, #, 8, Q String (one or more character) Hello, abc, b, Y (May be extended practically to different data types in programs (e.g. single, double, char, date, decimal etc.)

Variables and constants	Assign, identify and explain the use of constants and variables in algorithms and programs.	Assign, identify and use constants in programs and algorithms to store data that does not change. Assign, identify and use variables in programs and algorithms to store data that can change. Understand where constants and variables are appropriate and use appropriately.
	Describe the scope and lifetime of variables in algorithms and programs.	Scope of a variable refers to the visibility/accessibility of a variable. In other words, which parts of your program can see or use the variable. Can be global or local. Lifetime of a variable The lifetime of a variable is the duration of the location where the variable exists. Lifetime of Global variables. Global variables begin to exist when the program starts. Global variables cease to exist when the program exits. Lifetime of Local variables. A local variable begins to exist when the variable is defined within a subroutine. A local variable ceases to exist at the end of the scope in which the variable is defined (when the subroutine ends).

5. Security and authentication

Topic	Amplification	Teacher Guidance
Security techniques	Use appropriate security techniques, including validation and authentication.	Use techniques that validate data (could be any of the range of validation checks discussed) in algorithms. Use techniques that authenticate information entered into an algorithm.

Unit 3 Non-exam assessment (NEA)

Submission date

The work will be submitted so that it arrives with the moderator baccording to the examination timetable published for the year of academic certification. The task **must** be the one set by the WJEC as the task for that academic year.

Submission method

All marks must be submitted to WJEC's Internal Assessment Mark Input System (IAMIS). This system will automatically identify the required sample.

Work must be submitted in electronic format. Currently this will involve copying the sample to an online system on a per candidate basis which will distribute the work to the appropriate moderator. Work should be arranged as described on page 20 of the specification.

Practical implications of the rules of non-exam assessment

Conditions/ Resources

NEA must be conducted under the conditions prescribed. That is, accounts used for the NEA must be locked down between NEA sessions, and these accounts must have no access to the internet or email.

No work can be brought in from outside of the NEA controlled environment. Candidates are free to research outside of the controlled conditions, but this research should be used to develop their skills and knowledge. No typed work, diagrams or code should be brought into the exam. This rule is comparable to examination rules in that candidates are free to revise to improve knowledge and skills outside of examinations, but cannot bring in any physical work to an examination.

The time spent by candidates on the NEA must be tracked by centres and candidates must not be permitted to exceed the 20 hours. Centres are free to choose their method of tracking time, although there must be evidence of this. The WJEC may request this evidence at any time.

Release and practice tasks

Centres are free to use other tasks to ensure that candidates are upskilled in preparation for the task. This could be from specimen material or past papers.

Centres are free to issue 'practice tasks' prior to commencing the NEA to facilitate skill building. However these tasks must not be drawn directly from the released task.

Word count

The 2000 word limit relates to the report that is supplied with the programmed solution. This word count does not include the program code or annotation.

Time arrangements

The time permitted for the NEA is 20 hours. Clearly, it is impossible for candidates to work for 20 hours in a single sitting and therefore it is at the discretion of the centre of how to arrange this time.

The approaches below are valid; however centres may choose different methods.

1. Sitting the task in a single week

This would involve candidates working over a single week (say, for 2 x 2 hour sessions in a single day) and to produce the entire task over this time.

Advantages of this method

- The task is completed in a short period of time and this may improve manageability in terms of continuity
- Less susceptible to impact of learner absence
- The 'full immersion' achieved by this approach will result in focused, continuous work
- Students may research as homework nightly and this will be fresh in their memory on commencement of the next day's tasks
- Tracking of time spent is simple.

Disadvantages of this method

- Is highly reliant on all skills for the task having been taught prior to commencement of the task and is reliant on candidates remembering a large number of concepts
- Requires curriculum rearrangement
- Unforgiving if candidates become stuck on an individual area, as they will not be able to research and return to the task until they are released from the controlled conditions
- Inflexible if an individual candidate is absent during the task week

2. 'Chunking' the work over multiple sessions.

This approach would involve setting aside time within the curriculum for candidates and all work would be conducted during these sessions. (Sessions of say, blocks of 2 hours each week are run over a 10 week period).

Advantages of this method

- The task is completed in a defined time period and this may improve manageability in terms of tracking of time
- Skills for different sections can be taught discretely in lessons between test sessions
- Candidates may research as homework in the time between tasks.

Disadvantages of this method

- Requires curriculum rearrangement
- Candidates missing days where test sessions are being conducted will lose significant blocks of time and this will require rearrangement
- Candidates missing teaching sessions between NEA sessions will be under-skilled.

3. Running the task in normal lesson sessions

This would involve teaching skills relevant to the task and then switching to NEA conditions once candidates had been taught. For example, teaching the concepts of analysis for 30 minutes then switching to controlled conditions for the next 30 minutes once candidates had been taught the concepts required to tackle the task.

The lessons tasks **must not be drawn directly from the released task**, and only the skills required should be taught prior to commencement of the NEA.

Advantages of this method

- Does not require curriculum rearrangement
- Skills for individual sections can be taught discretely between test sessions.
- Candidates may research as homework in the time between tasks.

Disadvantages of this method

- Difficult to track time, as the time candidates are active on the NEA must be tracked in terms of the minutes spent (as the time on task may be a subset of a lesson (e.g. 30 minutes)
- The methodology can result in candidates relying on short term memory which will impact their understanding
- Fragmented delivery will impact candidates being able to see the bigger picture in the development of the project which may result in weaker or disjointed planning, implementation and evaluations.

Given Task

The task for Unit 3 will be a scenario set by WJEC and will take the form of a project that will be available for submission only in the year specified. A different scenario will be set for each academic year. All work carried out for this task should be under teacher supervision, with no access to the Internet or email. The time permitted for this task is 20 hours, and all time spent on the task should be monitored and logged by the centre as detailed in the specification.

The scenario will provide candidates with a description of a client's need for a new computer based solution to a given problem. In addition to a functional computer program, candidates will need to produce a word processed report with an advisory limit of 2000 words. The areas for inclusion in the report are covered in detail in the content for this component and are summarised below:

Section 1 – Scope of the problem

- Analysis of the given scenario in terms of input, processing and output
- Objectives, including measurable success criteria for the proposed system.

Section 2 – Design

Descriptions of:

- Input and output facilities required to produce a user interface
- Data structures that will be required
- Documentation of the following routines using a standard convention (pseudo code or flowchart):
    - validation routines
    - data handling and processing
    - authentication

Section 3 – Software development

- Annotated listing(s) of all programming code
- Evidence of the user interface

Section 4 – Test strategy

- Description of the test strategy
- Description of the purpose of unit, integration and functional testing
- Test plan and test data

Section 5 – Testing

- Evidence of implementation of the test plan and test outcomes with commentaries

Section 6 – Further development

- Discussion of the outcomes of the testing
- Description of the successful features of the solution and identification of areas for further development
- Suggestions for extensions to the solution

The Refinement Log

The refinement log is an integral part of the project and should be completed during each session. The purpose of the log is for candidates to demonstrate that they are working in a logical and systematic manner.

Candidates are expected to record any issues encountered and how these issues were addressed.

The refinement log is supplied as an electronic document and must be submitted with the computer program and the report.

A sample log page is shown on page 27 of the specification.

The marking grids should be used in conjunction with the advice given on page 28 of the specification. The NEA marking grids in the specification are designed in a vertical hierarchical table structure, however, in practice, these grids are easier to use in a horizontal format, as it is easier to compare the outcomes in the grids in this format. Below, the grids have been transferred to such a format to facilitate easier use.

Scope of the problem - AO3 Max 8 marks				
Band 0 **0 marks**	**Band 1** **1 - 2 marks**	**Band 2** **3 – 4 marks**	**Band 3** **5 - 6 marks**	**Band 4** **7 - 8 marks**
Response not credit worthy or not attempted	The candidate has: • Carried out a superficial analysis of the given scenario that has only partially identified the input, processes and output required to produce a working solution • Put forward objectives for the solution in terms of tasks to be carried out. Not all objectives are measureable or appropriate to the solution	The candidate has: • Carried out an analysis of the given scenario, identifying the basic: ○ data required to create a working solution ○ processing requirements to produce a working solution ○ outputs from the solution • Set objectives, a minority of which are measurable, that describe the main tasks required to create a working solution	The candidate has: • Completed an analysis of the given scenario, with no significant omissions, identifying most of the: ○ data required to create a functional solution ○ processing to be carried out by the solution ○ required outputs from the solution • Put forward objectives, most of which are measurable, that define the tasks to be carried out by the proposed solution	The candidate has: • Completed a thorough analysis of the given scenario identifying all: ○ data required to create an effective solution ○ processing to be carried out by the solution ○ required outputs from the solution • Produced a detailed set of objective, that are measurable, that define clearly the tasks required to create effective and fully functional solution

Design - AO3 Max 12 marks				
Band 0 **0 marks**	**Band 1** **1 - 3 marks**	**Band 2** **4 - 6 marks**	**Band 3** **7 - 9 marks**	**Band 4** **10 - 12 marks**
Response not credit worthy or not attempted	The candidate has: • Used the objectives for the solution as a basis for an outline design for a partial solution • Produced outline designs for the identified input and output facilities provided by the user interface • Outlined the key files and/or data structures required to produce a partial solution • Given consideration to possible validation of input data, which may not be accurate or appropriate • Partially outlined processing routines that may use a standard convention such as pseudo code or flowcharts. The descriptions may not be accurate or correct.	The candidate has: • Used the objectives for the solution as a basis for the design that will produce a solution that will achieve a majority of the required functionality • Identified the basic input and output facilities to be provided by the user interface • Identified the data structures required to produce a solution that is partially functional but carries out the basic requirements of the given scenario • Identified several inputs that will require validation • Outlined the need for authentication processes • Described the basic processing routines for the solution as algorithms using a standard convention such as pseudo code or flowcharts. Some routines may be incorrect or incomplete.	The candidate has: • Used the objectives for the solution to inform a design that will produce the facilities required to ensure that the solution is functional • Identified and described most of the input and output facilities to be provided by the user interface and has considered the needs of the user • Described most data structures required using appropriate terminology • Identified most inputs that will require validation and outlined proposals for implementing validation routines • Considered the need for authentication routines • Described most processing routines for the solution as algorithms using a standard convention such as pseudo code or flowcharts	The candidate has: • Produced a comprehensive design that would allow a competent third party to create a solution that covers all stated objectives • Identified fully and described in detail the input and output facilities to be provided by the user interface which will be fit for purpose • Described all data structures required to create an effective solution, using correct technical terminology • Described fully the validation routines required to ensure that only appropriate data can be entered into the solution • Considered fully the need for authentication • Described all processing routines for an effective solution as algorithms using a standard convention such as pseudo code or flowcharts

Refinement Log - AO3 Max 5 marks			
Band 0 **0 marks**	**Band 1** **1 mark**	**Band 2** **2 - 3 marks**	**Band 3** **4 - 5 marks**
Response not credit worthy or not attempted	The candidate has: • Outlined the progress made in most sessions • Described problems encountered but may lack the use of technical terminology • Outlined changes that have been made to the original design • Identified one or more activity for the next session	The candidate has: • Demonstrated a structured approach to developing the solution • Carried out activities in an appropriate order • Described the progress made in each session • Provided a description of any problems encountered with satisfactory use of technical terminology • Described any changes that have been made to the original design • Produced sensible actions for subsequent sessions	The candidate has: • Demonstrated a structured approach to developing the solution • Carried out activities in an appropriate order • Evaluated effectively the progress made in each session • Provided a full description of any problems encountered with good use of technical terminology • Justified any changes that have been made to the original design demonstrating an informed understanding of the need for change • Produced logical and prioritised actions for subsequent sessions

Effectiveness of solution – AO3 Max 15 Marks					
Band 0 **0 Marks**	**Band 1** **1 – 3 Marks**	**Band 2** **4 – 6 Marks**	**Band 3** **7 – 9 Marks**	**Band 4** **10 – 12 Marks**	**Band 5** **13 – 15 Marks**
Response not credit worthy or not attempted	The candidate has created a solution that: • Partially achieves the requirements of the given scenario • Has a partially functional user interface	The candidate has created a solution that: • Achieves the basic requirements of the given scenario • Includes a basic user interface that may not be fit for purpose • Includes a basic number of elements of structured programming	The candidate has created a solution that: • Achieves a majority of the requirements of the given scenario • Provides evidence of a functional user interface • Includes some elements of structured programming • Makes use of authentication routines • Is portable	The candidate has created a solution that is: • Correct and fulfils most of the requirements of the given scenario • Usable with a user interface that is functional and generally easy to use • Structured and modular in nature • Secure with authentication routines • Portable and generally robust	The candidate has created a solution that is: • Correct and fulfils all the requirements of the given scenario • Usable with a user interface that is intuitive and fit for audience and purpose • Well-structured and modular in nature • Efficient in use of resources • Secure with effective authentication routines • Portable, reliable and robust

Technical Quality – AO3 Max 20 Marks					
Band 0 **0 Marks**	**Band 1** **1 – 4 Marks**	**Band 2** **5 – 8 Marks**	**Band 3** **9 – 12 Marks**	**Band 4** **13 – 16 Marks**	**Band 5** **17 – 20 Marks**
Response not credit worthy or not attempted	The candidate has created a partial solution to at least one of the requirements of the given scenario. The candidate has: • Made use of meaningful identifiers • Used indentation	The candidate has created a solution that covers the basic requirements of the given scenario. The candidate has: • Written code that includes some self-documentation • Used appropriate indentation • Made use of meaningful identifiers • Provided basic annotation of the code	The candidate has created a functional solution that covers the majority of the requirements of the given scenario. The candidate has: • Written code that is self-documenting • Used a consistent programming style including indentation • Made use of meaningful identifiers and appropriate use of constants • Made use of local variables and generally minimised the use of global variables • Produced validation routines • Provided annotation of the code where appropriate	The candidate has created a functional solution that covers most requirements of the given scenario. The candidate has: • Written code that is self-documenting and modular in nature • Used a consistent programming style including indentation • Made use of meaningful identifiers and appropriate use of constants • Made use of local variables and minimised the use of global variables • Produced validation routines and created routines for exception handling • Provided effective annotation of the code where appropriate	The candidate has created a successful solution that covers all the requirements of the given scenario. The candidate has: • Written code that is self-documenting, well-structured and modular in nature • Used a consistent programming style throughout, including indentation and the use of white space around operators and keywords • Made full use of meaningful identifiers and appropriate use of constants • Created subroutines with well-defined interfaces • Made effective use of local variables and minimised the use of global variables • Produced effective validation routines and created routines for exception handling • Provided informed annotation of the code where appropriate

Test strategy - AO2 Max 8 marks			
Band 0 **0 marks**	**Band 1** **1 - 2 marks**	**Band 2** **3 - 5 marks**	**Band 3** **6 - 8 marks**
Response not credit worthy or not attempted	The candidate has: • Attempted to describe of the scope and range of the chosen test strategy • Identified the purpose of testing the solution • Produced a test plan to carry out the testing of the solution but the plan may not cover all key areas • Identified test data to partially test the solution	The candidate has: • Considered the nature of the solution when developing a test strategy • Provided a description of the scope and range of the chosen test strategy • Described the purpose of unit, integration and functional testing, taking into account the nature of the solution • Considered how the outcomes of the testing process may be useful in any further development of the solution • Produced a test plan to carry out unit, integration and functional testing of most of the requirements of the given scenario • Identified appropriate test data to test most functionality of the solution	The candidate has: • Considered fully the nature of the solution when developing a well-structured test strategy • Provided an informed description of the scope and range of the chosen test strategy • Fully explained the purpose of unit, integration and functional testing, taking into account the nature of the solution • Considered in detail how the outcomes of the testing process will be used to influence any further development of the solution • Produced a comprehensive plan for carrying out unit, integration and functional testing to cover all requirements of the given scenario • Identified comprehensive test data to fully test the solution

Testing - AO3 Max 8 marks			
Band 0 **0 marks**	**Band 1** **1 - 2 marks**	**Band 2** **3 - 5 marks**	**Band 3** **6 - 8 marks**
Response not credit worthy or not attempted	The candidate has: • Made use of data to test most areas of the solution that have been completed • Presented testing outcomes with brief correct commentaries	The candidate has: • Used the test plan to carry out testing of the solution • Made use of realistic data to test all areas of the solution • Presented testing outcomes with suitably technical commentaries	The candidate has: • Followed the test plan in a logical and systematic manner • Made effective use of typical, extreme and erroneous data • Presented all testing outcomes with detailed and informed commentaries

Further development - AO3 Max 4 marks			
Band 0 **0 marks**	**Band 1** **1 mark**	**Band 2** **2 - 3 marks**	**Band 3** **4 marks**
Response not credit worthy or not attempted	The candidate has outlined: • Outcomes of the testing process • Suggestions for extension to the solution	The candidate has: • Considered the outcomes of the testing processes against the solution objectives • Identified successful features and areas for further development • Proposed specific suggestions for extensions to the solution	The candidate has: • Considered fully the outcomes of the testing process in terms of the solution objectives • Fully described the successful features and areas for further development • Proposed detailed and comprehensive suggestions for specific extensions to the solution