'I will take responsibility for my learning, be intellectually curious and work independently at school and at home.'

The Regis School
The best in everyone™
Part of United Learning

Computer Science

EXAM BOARD: OCR **COURSE CODE: 277**

TOPIC NUMBER	TOPIC
01: Computer Systems	
1.1	Systems Architecture
1.2	Memory and Storage
1.3	Computer Networks, Connections and Protocols
1.4	Network Security
1.5	Systems Software
1.6	Ethical, legal, cultural and environmental impacts of digital technology
02: Computational Thinking, Algorithms and Programming	
2.1	Algorithms
2.2	Programming Fundamentals
2.3	Producing Robust Programs
2.4	Boolean logic
2.5	Programming languages and Integrated Development Environments
Checklists	Knowitall Ninja.com / Codeacademy / Edulite
Code	OCR Exam Reference Language
Key Terms	Definitions

Name: .....

Tutor Group: .....

September 2021

Common CPU Components

ALU (Arithmetic Logic Unit)

- Performs arithmetic calculations (add, subtract, multiply)
- Performs logic operations (e.g. <, >, =, !=).

CU (Control Unit)

- Controls the flow of data between registers in the CPU.
- Controls input and output of data to and from the CPU
- Controls the timing of signals sent within the CPU.

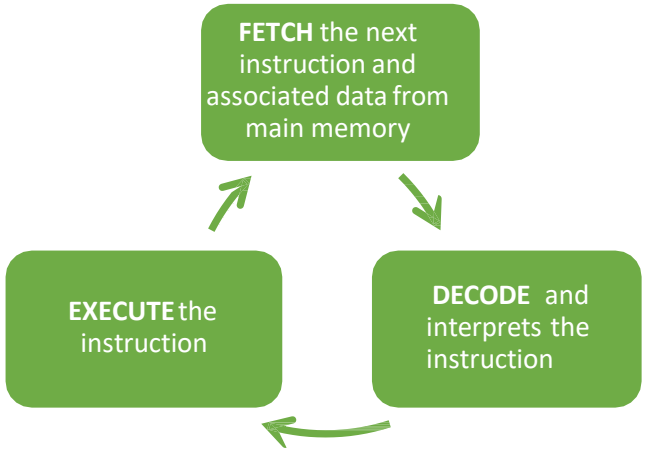
CACHE

- Stores frequently used instructions and data
- Built onto the CPU and so provides quicker access than RAM
- Allows instructions and data to be loaded into the CPU more quickly.

Registers

- Very fast memory on the CPU itself
  - PC (Program Counter) - stores the address of the next instruction
  - MAR (Memory Address Register) - stores the address of the next instruction to be accessed
  - MDR (Memory Data Register) – stores the data to be brought from or sent to main memory
  - ACC (Accumulator) – stores the value currently being worked on

The Fetch – Execute Cycle



The Purpose of the CPU

- The CPU is the brain of the system.
- Processes all the data and instructions to make the system work.
- It is installed on the motherboard.

1.1 Systems Architecture

The Purpose of Embedded Systems

- To provide a specific, pre-defined function
- Cheaper than providing a full personal computer system.
- Can be made much smaller than a personal computer system
- Allows for a device to be automated / programmed.

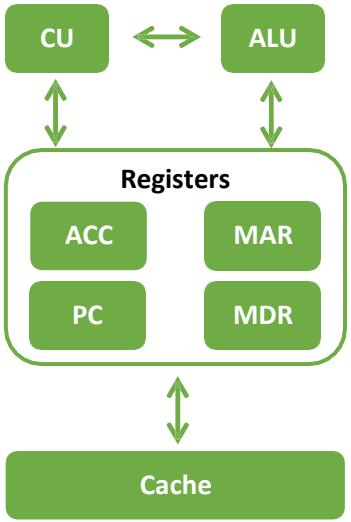
The Characteristics of Embedded Systems

- Task specific.
- The task is performed in a certain time frame.
- Do the same thing repeatedly.
- Basic or no UI.
- May respond to sensors.

Examples of Embedded Systems

- Dishwasher
- MP3 player
- Washing machine
- Mobile phone
- Manufacturing equipment
- Tills

Von Neuman Architecture



How Common Characteristics of CPUs Affect Performance

Clock Speed

- A faster clock speed allows more instructions carried out per second and so instructions are executed more quickly.
- This allows for more programs to be run at the same time.
- This also allows for more complex processing operations to be completed in real time.

Cache Size

- A larger cache gives more space for frequently used instructions.
- This provides more storage for fast access, meaning faster fetching of instructions and so faster processing

Number of Cores

- More cores allow more instructions carried out simultaneously.
- More cores allow the processor to process more instructions at the same time.
- This allows batches of instructions to be executed more quickly, which allows for more programs to be run at the same time.

Character Sets

- Allow computers to understand letters, numbers, and other characters
- Logically ordered, the value for A is lower than B
- ASCII
  - American Standard Code for Information Interchange
  - Each character is given a unique binary code
  - A = 065 = 01000001
  - Code is 8 bits (1 byte) long
  - 256 possible characters
- Unicode
  - Uses 2 bytes giving many more characters.
  - Accommodates languages such as Arabic with thousands of characters

Storing Sound

- Computers only understand binary so sound must be encoded
- Broken down into thousands of samples per second, each is stored as binary data
- Sample rate
  - Measured in Hz
  - How many samples per second
  - more samples = more detail = clearer sound = more space needed
- Bit Depth
  - Number of bits available for each sample
  - Higher bit depth = higher quality = more space needed
- Duration
  - Higher duration = longer audio = more space needed

Storing Images

- Images are stored as a series of pixels in binary
- Each pixel has a specific colour, represented by a specific code
- Also contains metadata
  - Structure of the file
  - Size of the grid
  - Other info such as date
- Resolution is the number of pixels in the image
  - Higher resolution = more pixels = clearer image = more space needed
- Colour depth is the number of bits used to store the colour for each pixel
  - 1 bit allows 2 values, 2 bits allow 4 values etc.
  - Higher colour depth = more realistic colours = more space needed

Compression

- Encoding data so that is needs fewer bits/bytes to represent it
- Reduces space needed for storage
- Must be decompressed to be used
- Lossless Compression
  - Compresses data files without losing any of the information
  - Reversible - that the original data can be reconstructed
  - Not all files can be compressed in this way
- Lossy Compression
  - Does lose some of the information
  - Used where this is acceptable e.g. audio
  - Produces smaller files
  - Poorer accuracy

Units of Data Storage

- Bit
- Nibble - 4 bits
- Byte - 8 bits
- Kilobyte (KB) - 1,000 bytes
- Megabyte (MB) - 1,000 KB
- Gigabyte (GB) - 1,000 MB
- Terabyte (TB) - 1,000 GB
- Petabyte (PB) - 1,000 TB

Converting Binary to Hex

8	4	2	1
0	1	1	0

4 + 2 = 6

6

8	4	2	1
1	1	0	1

8 + 4 +1 = 13

D

Converting Between Denary and Binary

128	64	32	16	8	4	2	1
1	0	0	0	1	0	1	1

Converting Denary to Hexadecimal

62 ÷ 16 = **3** R 14
3 ÷ 16 = **0** R 3
3 14
3 E

Binary

- Binary is a number system made up of 1s and 0s
- There are only two possibilities, so this is a base two number system
- Computers use binary because the CPU contains transistors, which are either on or /off

Hexadeci

- Hexadecimal is a number system using 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
- There are 16 possibilities, so this is a base sixteen number system
- Binary strings are long and difficult to work with. Hex is shorter
- Hex is easily converted to binary as there is 1 hex digit per nibble.
- Hex is less prone to error

Choosing Storage Media

When choosing a storage media there are several factors to consider:

Capacity
- How much data the storage media can hold
- Larger files such as videos/ music/ pictures will require larger amounts of storage

Speed
- How quickly the data can be written and read back
- Some situations, such as a live website, will need data to be accessible quickly
- In other situations, such as a backup, it is acceptable for the process to take longer

Portability
- How easy it is to move the storage media from one device to another.
- The size of the media itself and the compatibly of the media

Durability
- The length of time the storage media is expected to last
- How easily damaged the storage media is

Reliability
- How likely the storage media is to fail and how likely errors are to occur.

Cost
- How expensive the storage media and any required hardware is.

1.2 Memory and Storage

Primary Storage

- Much faster than secondary storage
- Holds data and instructions needed by the

ROM (Read Only Memory)

- Values stored in ROM remain when the computer is switched off (non volatile).
- Virus attacks are unlikely.
- Values stored cannot be accidentally changed.
- Data is written permanently when the computer is built.
- Holds the instructions for booting the computer.

RAM (Random Access Memory)

- Loses its data when the computer is switched off (volatile)
- Used to save data about programs that are currently open.
- Much faster than a HDD or SSD, and so the CPU has to spend less time in the "fetch" part of the "fetch-decode-execute".
- It is more expensive per GB than a HDD or SSD. This limits our usage of RAM, and the amount that can be installed.

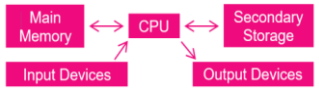
Virtual Memory

- Virtual memory is simulated memory that is written to a file on the hard drive.
- It lets more memory be used than there is in the system.
- This is useful when you need to run more applications on the computer than RAM can support.

Virtual Memory Implementation

1. The Operating system sets up virtual memory using the Virtual Memory Manager (VMM).
2. The VMM creates a file on the hard disk large enough for the extra memory needed.
3. The Operating System can then address the virtual memory as if it were real memory stored in RAM.
4. Swapping or paging is the process used by the operating system to move data between RAM and virtual memory.
5. Data which processes do not need immediately is moved out of the RAM to virtual memory.

Memory Data Flow



Calculating File Sizes

- sound file size = sample rate x duration (s) x bit depth
- image file size = colour depth x image height (px) x image width (px)
- text file size = bits per character x number of characters

Secondary Storage

- Storage devices which are not constantly connected to the computer.
- Storage devices not directly accessible by the system's CPU.
- Used to back up data stored in primary storage.
- Useful when there is a need for larger storage capacity.

Types of Secondary Storage

Optical Storage
- Data is written and read using a laser beam
- Examples include CDs and DVDs
- Inexpensive, reliable, robust, relatively large capacity

Magnetic Storage
- Uses different magnetic patterns to store data
- Examples include tape cartridge and hard drive
- Large capacity, can be used to store operating system and other files and programs, reliable, cost-effective

Solid State Storage
- Data is stored within flash chips
- Examples include USB drives and SSDs
- Flexible, faster access to data, can be used for portable devices, generally smaller in size, robust, easy to use

Examples of Choosing Storage Media

A portable barcode scanner uses solid state flash media

- **Capacity:** barcodes do not consume much data so high capacity is not required.
- **Speed:** flash media is quick and delays when scanning would affect the device's operation
- **Portability:** flash media is small and light and so will easily fit within the scanner
- **Durability**: flash media has no moving parts so the device can be moved without damage
- **Reliability**: flash media is highly reliable
- **Cost:** flash media is more expensive than other storage, but this is outweighed by the above factors

Films are sold on DVD and BluRay disks

- **Capacity:** high capacity to allow for longer, higher quality movies
- **Speed:** slower access speeds than flash memory but sufficient for the task
- **Portability:** lightweight, small and commonly used
- **Durability**: durable if stored correctly
- **Reliability**: reliable if stored correctly
- **Cost:** very cheap to produce in high quantities

Factors Which Affect Network Performance

Bandwidth is the amount of data that can be transmitted. The larger the amount of available bandwidth, the more data which can be transmitted in a period oftime.

- **The type of connection** – wired connection will be faster than wireless.
- **Interference** – walls and other radio signals can interfere with wireless networks. Electrical cables can interfere with wired networks.
- **The number of devices** – if lots of devices are using the network, there will be less bandwidth available to each user.
- **The type of media being accessed** – large files consume more bandwidth, large files such as HD video will take longer to transfer.

If insufficient bandwidth is available for the number of users, or the size of files, performance will be poor.

Protocols

- **HTTP** – HyperText Transfer Protocol – Web pages
- **HTTPS** - Hypertext Transfer Protocol (Secure) – Secure web pages
- **FTP** – File Transfer Protocol
- **SMTP** – Simple Mail Transfer Protocol – Send emails
- **IMAP** – Internet Message Access Protocol – Receive emails
- **POP3** – Post Office Protocol version 3 – Receive emails
- **DNS** – Domain Name System – Converts names to IP Addresses
- **IP** – Internet Protocol – Addresses packets
- **TCP** – Transmission Control Protocol – Provides reliable transmission

The Internet

The Internet – A worldwide collection of computer networks
Hosting – A service which allows you to publish a website to The Internet **DNS (Domain Name System)** – A system for converting host names and web addresses into IP addresses
Web Server – A server configured to host websites.
Web Client – A client accessing websites, usually over The Internet.

Addressing

- Addressing allows us to identify devices
- Every device has a MAC address which never changes
- Each device on a network has an IP address but this can change

MAC Address

- Media Access Control (MAC) - 48 bits.
- Hexadecimal values
- MM:MM:MM:UU:UU:UU - MM is the manufacturer ID and UU is the device ID.

IP Address

- 32 bits using 4 sets of decimal values from 0 – 255.
- Used to route traffic to the right network.

Advantages Of Networking Computers

- Easy to share documents, several people can work on a document at once.
- Only one Internet connection is needed and can be shared between devices.
- Centralised backups can be carried out.
- Software updates and patches can be automatically pushed out.
- Users can log in to any machine connected to the network.

Network Hardware

Wireless Access Points
Converts network signals into radio waves allowing devices to connect wirelessly.
Routers and Switches
Connects devices on a LAN together by transmitting data between devices.
NIC (Network Interface Card)
A piece of hardware within a device which allows it to connect to the network.
Transmission Media
Connects the NIC to the router or switch. Could be:

- Wireless - using radio waves
- Ethernet – twisted pair coper cables
- Fibre Optic – data transmitted as light through glass or plastic cable

1.3 – Computer Networks, Connections and Protocols

Encryption

- A method of scrambling data with a key.
- Anyone can join an open Wi-Fi network and see traffic from other users.
- If encrypted data is intercepted, it will have no meaning.
- To read the data, the user must decrypt it using the key.
- The encryption method used is called 'SSL' (Secure Socket Layer).

Cloud Computing

Services such as software, processing or storage hosted in a remote location and accessed via The Internet.

- Easy and quick to increase or decrease resources.
- Maintenance is performed by the cloud provider.
- Data is stored away from the organisation's building.
- There is no upfront cost, organisations pay only for what they use each month.
- Relies on having a suitable Internet connection

Layers

- In a network, data travels through layers where protocols add or removing extra information.
- Layers allow one part of the protocol to be changed or rewritten without affecting the other parts
- Consistency of communication components – ensures that different hardware and software can communicate.
- Divides communication into smaller components - makes troubleshooting easier.

Modes of Connection

Ethernet

- For communication over a wired network.
- Uses a Media Access Control (MAC) address.
- Uses error checking.
- Devices check that no other device is communicating over the link before sending.

WiFi

- Wireless connection which uses radio waves to transmit data through the air.
- Uses an SSID to identify the network.
- Uses WPA2 or WEP to encrypt and secure data.
- Unsecured traffic can be intercepted easily.

Bluetooth

- Wireless connection which uses radio waves to transmit data through the air.
- Much shorter range than Wi-Fi
- Usually used for a direct connection between two devices.
- Bluetooth headphones, mice and keyboards are very common.
- It is possible to send files using Bluetooth but this is slow.

Types Of Network

LAN (Local Area Network)

- Covers a small geographical area
- Usually contained within one building
- Equipment is owned by the organisation
- Lower setup costs
- Faster speeds
- More control over security

WAN (Wide Area Network)

- Covers a large geographical area
- Connects buildings, towns or cities together
- Equipment is owned by a telecommunications company
- Higher setup costs
- Lower Speeds
- Less control over security

Client-Server Networks

- All devices are connected to a server.
- The server stores user account details.
- Clients access services from servers.
- Servers receive and processes requests from clients.
- File servers, web servers, database servers etc. all provide different services
- If the server fails, clients will be unable to operate.
- They are more involved to setup.

Peer to Peer Networks

- All devices have equal status.
- There is no central server, making them relatively easy to maintain.
- If one device fails only the information stored on that device will be inaccessible but the network will still operate.
- They are relatively easy to set up.
- There is no central control, making security and administration harder.

Star Network

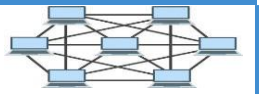
Advantages

- If a single link breaks the network still stays active.
- If one connection fails it does not affect the rest of the network.
- Easy to add additional devices onto the network.
- Fast because each device has its own connection to the switch / server.
- There are few data collisions.

Disadvantages

- Dependent on one central device.
- If the central device fails, the whole network fails.
- The performance of the network is dependent upon the central device.
- The number of devices is restricted by the central device.

Mesh Network



Advantages

- If a link breaks another route is available.
- The fastest route can be chosen.
- Can be quite cheap if wireless.

Disadvantages

- Expensive if wired.
- More complicated to maintain
- Set-up and maintenance can be costly

Forms of Attack	
Active	An attempt to modify or delete data, or to stop the network from operating correctly.
Passive (Eavesdropping)	An attempt to find information about the network or retrieve information without changing anything.
Internal	An attack by someone inside the organisation.
External	An attack by someone outside the organisation.

Qualities of a Strong Password

- At least eight characters
- Include upper case and lower case
- Include special characters
- Include numbers
- Does not include a name
- Does not contain a complete word
- Relates to an acronym

Types of Malware	
Viruses	Malicious software hiding within another application. Designed to harm a network or computer system
Worms	Similar to viruses but not hidden within other files. Replicates through a network to spread to other computers
Trojans	Programs which pretend to be legitimate but are malware. Often disguised as email attachments. Cannot spread by themselves and so deceive a user into installing them.
Spyware	Monitors user activities and send the information back to an attacker.
Ransomware	Blackmails users into making a payment to an attacker. Some will only try to frighten users into paying, others will encrypt files

1.4 - Network Security

Threats to a Network	
Social Engineering	Where users do not follow policies, make a mistake such as using their name as password, or are tricked into giving out information. Phishing emails trick users into giving away information. Pretends to be a genuine message with a link to a website that looks like the real company.
Brute force	Trial and error. Tries all possible passwords until the correct one is found.
Denial of service (DOS)	Overloads a computer or network with traffic by bombarding it with requests.
Data interception and theft	Looking at data travelling over a network, often using software called a packet sniffer.
Structured query language (SQL) injection	Affects websites which use a SQL database. SQL code is entered into a data input field on the website to look at or modify data stored in the database.
Malware	Malicious software designed to cause harm to a system or network. Users are often tricked into running malware.

Identifying and Preventing Vulnerabilities

- **Penetration testing** - The network is scanned for security weaknesses, vulnerabilities and poor configuration to find problems before an attacker can. Software is often used to automate this process. Allows organisations to find and fix threats before attackers can use them.
- **User access levels -** Controls which parts of a system users can access. Users should only be given access to parts of a system they need. Limits the actions a user can take. Reduces the risk of both deliberate data theft, but also the damage that can be cause by social engineering attacks or malware.
- **Secure passwords -** Passwords should not be easy to guess, should be long and include numbers and symbols. Passwords should not be shared. Defence against brute force attacks, these take much longer with secure passwords.
- **Encryption -** Data is translated into code so that only those with the key can read it. Means that if data is intercepted it cannot be read. Defence against data interception and theft.
- **Anti-malware Software -** Prevents malware from being installed and removes any that is installed. Includes anti-virus software, anti-phishing tools and anti-spyware software. Scans all the files on a computer and checking them against a list of known malware.
- **Firewalls -** Monitors traffic going into and out of a computer or network, and either allows or blocks it. Forms a barrier between a system and the attacker.
- **Physical Security –** Controls access to servers, networking equipment and other important hardware. May take the form of security guards, locks, CCTV or swipe cards.

Operating Systems

- Tells the hardware what to do.
- Allows the computer to run other applications.
- Controls the operations of a computer.
- Provides an interface between the computer and a user.
- Hides the complexities of the hardware from the user.

Examples include:

- Microsoft Windows
- Apple OS X
- Linux
- Android
- Apple IOS

Operating System Features

User Management

- Individual users can be created and deleted.
- Allows more than one person to use a computer with their own files and settings.
- Access levels control user access to systems for security.
- A log is kept of files a user creates, accesses, edits and deletes.

File Management

- All files are given a name.
- Files are stored in folders.
- Users can create, modify, move, and delete files and folders.
- Users can sort or search for files and folders.
- Users can restore deleted files.
- Users can set access rights to files.

Multitasking

- Many tasks can be executed on a computer simultaneously.
- An operating system has several processes running at the same time.
- Processor is given a small part of each task one after the other.
- All tasks appear to be executing at the same time.
- In reality, resources are shared between tasks.

Peripheral Management

- Allows devices to communicate with the computer.
- Data is transferred between devices and the processor.
- Controlled using Device Drivers:
  - Contain instructions on how to control a device.
  - Each device has its own driver.
  - Any device can be used if a driver is available.
  - Drivers can be updated to give better performance or fix bugs.

1.5 – Systems Software

| Interface between the hardware and applications | Programs that run application software. | Software that helps the computer to run |

User Interfaces

- Allow the user to interact with the computer in a visual way.
- Graphs, text or audio are presented to the user.

GUI

- A type of interface that uses Windows Icons Menus and Pointers (WIMP) to represent the interaction between the user and a computer.
- Users use a mouse to interact with features displayed on the monitor.
- Powerful and easy to use but require a lot of processing power.

CLI

- User types a text command using the keyboard.
- The computer displays the results on the monitor.
- Requires little processing power and is extremely powerful.

Memory Management

- The management and organisation of memory at the system level.
- Memory is allocated between the different programs which are open.
- Programmers and users need not know where in memory data is held.
- Allocates free memory to programs that need it
- Frees up memory which is no longer needed.
- Controls the computer's memory to optimise performance.

Utility Software

Defragmentation Software

- Files on a disk are broken down into a series of segments.
- When files are deleted, the segments where they were stored are made available for new files.
- The new file may need more segments than the old, and so the segments allocated to it are not together on the disk. This is known as fragmentation.
- A fragmented disk takes longer to read from and write to, making the computer slower.
- Defragmentation software rearranges the segments so that they are stored next to each other.
- This reduces read/write time and improves performance.

Encryption Software

- Scrambles the contents of files so they can only be understood by authorised users.
- A key or secret code is needed to descramble the content.
- Takes longer for messages to be sent and received.
- Increases security as data cannot be read if stolen.
- Can encrypt individual files or the whole hard disk.

Data Compression

- Reduces the size of a file.
- Uses algorithms.
- Smaller files are easier to transmit.
- Allows more files to be stored in the same space.
- Lossless - no data is lost, and the original can be recreated.
- Lossy - some data is lost and the original file cannot be recreated.

1.6 – Ethical, Legal, Cultural and Environmental Impacts of Digital Technology

Ethical Issues

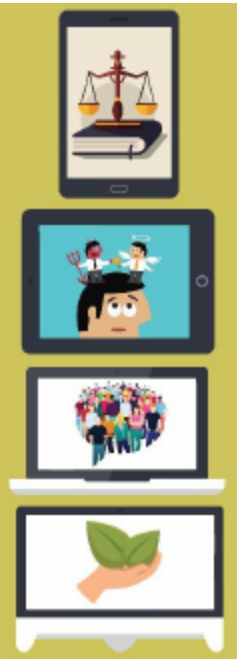
- Ensuring public safety.
- Cyber bullying.
- Unequal access to materials.
- The Digital divide.
- Virtual Currencies.
- Social pressure to be online and keep up with the latest technology.
- Access to inappropriate / illegal content.

Environmental Issues

Positive	Negative
• Industries such as manufacturing and agriculture are becoming more efficient • Increase in renewable energy options. • Electronic communication reduces the need to travel.	• Extraction of natural resources depletes them. • Electronic components require precious metals • Devices need large amounts of energy • Large amounts of e-waste • People want the latest devices, causing old devices to go to waste

Legal Issues

- Illegally sharing personal data
- Stealing money or information
- Illegally copying and sharing films and music
- Extorting information or blackmailing
- Electronic Spying



Cultural Issues

- Automation can improve the production process at the cost of jobs
- Technology has allowed jobs to be moved abroad where costs are lower
- Not everyone is proficient with technology
- Not everyone can afford technology
- Internet access may be poor in rural areas

Privacy Issues

- Devices may be tracked
- Social media encourages people to post about themselves online
- Unwanted images and people may be put online
- Big data allows information from many different sources to be put together
- Electronic information can be more easily copied
- Once information is online it is very difficult to remove it
- Not everyone is aware how to correctly use privacy settings

The Data Protection Act 2018

Personal data must:

- Data must be collected and used fairly.
- Data must only be held and used for the reasons which it was gathered.
- Data can only be used for registered purposes .
- Data held must be adequate, relevant and not excessive.
- Data must be accurate and up to date.
- Data cannot be kept for longer than necessary.
- Data must be kept safely and securely.
- Data cannot be transferred outside of the EU unless suitable laws are in place.

Key roles:

- **Information Commissioner** has overall responsibility for enforcing the Data Protection Act.
- **Data Controller** The person or organisation responsible for the data
- **Data Subject** The person who's data is collected

Copyright Designs and Patents Act 1988

- Gives creators of digital media the right to control how their work is used and distributed.
- Music, books, videos, games and software are covered by the act.
- Anything you design or code is automatically copyrighted and may not be copied without permission.

Software Licenses

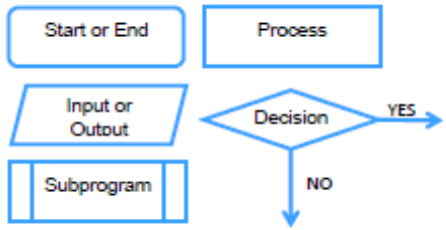
Open Source	Proprietary
• Free and available to anyone • Can be modified to suit different needs • Encourages collaboration • Quick to fix issues • Can include more bugs • Less secure • No official support	• Licence can be expensive • Support from the manufacturer • Usually more secure • Bugs issues fixed regularly • Usually has user documentation • Cannot modify the code • Copyrighted by a company or owner

Computer Misuse Act 1990

- It is illegal to access data stored on a computer unless you have permission to do so.
- It is illegal to access data on a computer when that data will be used to commit further illegal activity, such as fraud or blackmail.
- It is illegal to make changes to any data stored on a computer without permission. This includes installing a virus or other malware which damages or changes the way the computer works.
- The maximum punishment for breaking this law is a £5,000 fine or several years' imprisonment.
- It must be proved that access was intentional, and not accidental as a result of poor configuration

Flowcharts

- Created to represent an algorithm.
- Show the data that is input, and output.
- Show processes that take place.
- Show any decisions and repetitions that take place.
- Lines show flow through the chart.
- Shapes represent different functions



Searching Algorithms

Linear Search

1. Check the first value
2. If it is desired value
   - Stop
3. Otherwise check the second value
4. Keep Going until all elements have been checked or the value is found

Binary Search

1) Put the list in order.
2) Take the middle value.
3) Compare it to the desired value.
   a) If it is the desired value.
      i) Stop.
   b) If it is larger than the desired value.
      i) Take the list to the left of the middle value.
   c) If it is smaller than the desired value.
      i) Take the list to the right of the middle value.
4) Repeat step 3 with the new list.

Trace Tables

- Tests algorithms for logic errors which occur when the algorithm is executed.
- Simulates the steps of algorithm.
- Each stage is executed one at a time allowing inputs, outputs, variables, and processes to be checked for the correct value at each stage.

```
X = 3
Y = 1
while X > 0
      Y = Y + 1
      X = X - 1
print(Y)
```

Stage	X	Y	Output
1	3	1	
2		2	
3	2		
4		3	
5	1		
6		4	
7	0		
8			4

2.1 Algorithms

Key Concepts

- Computational thinking:
  - The use of computers to solve problems.
  - Development of algorithms to solve problems.
  - Using abstraction, decomposition, and algorithmic thinking.

- Abstraction
  - Using symbols and variables. to represent a real-world problem with a computer program.
  - Removing unnecessary elements
  - Example - a program is to be created to let users play chess against the computer.
    - Board is created as an array(s).
    - Pieces are objects that have positions on the board
    - The shape and style of the pieces may not be required.

- Decomposition
  - Breaking down large problems into a set of smaller parts.
    - Smaller problems are easier to solve
    - Each part can be solved independently
    - Each part can be tested independently
    - The parts are combined to produce the full problem.
  - There are usually several different approaches, and not one single right way to do this.

Sorting Algorithms

Bubble Sort

1) Take the first element and second element
2) Compare the two
   a) If element 1 > element 2
      i) Swap then
   b) Otherwise
      i) Do nothing
   c) Move to the next pair in the list
   d) If there are no more elements return to step (1)
   e) Otherwise, return to step (2)
3) Repeat until you have worked through the whole list without making any changes

Merge Sort

1) Split the list into individual elements.
2) Merge the elements together in pairs, putting the smallest element first.
3) Merge two pairs together, putting the smallest first.
4) Keep merging until all pairs are in order.

Insertion Sort

1) Element 1 is a sorted list.
2) The remaining elements are an 'unsorted' list.
   a) Compare the first element in the 'unsorted' list to each element in the sorted list.
   b) If it is smaller, put it in in front of that element and move the others along.
   c) If it is larger compare it with the next element.
   d) Keep going until you reach the end of the list, at this point put it in the final position.
3) Repeat the above until all elements have been sorted

Exam Reference Language

- Looks like pretend code.
- A more formal way to represent an algorithm for the exam.
- More like a programming language but does not compile.
- Is easy for programmers to read.

```
mark = input("Input mark")
if mark < 50 then
      print("Fail")
elseif mark < 70 then
      print("Pass")
elseif mark < 90 then
      print("Merit")
else
      print("Distinction")
endif
```

Pseudocode

- Uses short English words and statements to describe an algorithm.
- Generally looks a little more structured than normal English sentences.
- Flexible.
- Less precise than a programming language.

```
IF Age is equal to 14 THEN
Stand up
ELSE Age is equal to 15 THEN
Clap
ELSE Age is equal to 16 THEN
Sing a song
ELSE
Sit on the floor
END
```

Completing An Algorithm

1. Read what the algorithm should do.
2. Note down the steps that should take place.
3. Read the steps of the algorithm you already have.
4. Use your notes to write code to fill in the gaps.

Correcting An Algorithm

1. Read what the algorithm should do.
2. Note down the steps that should take place.
3. Read each step of the algorithm.
4. At each step, compare what the algorithm does to your notes about what it should do.
5. Take action to correct the algorithm where it differs from your notes.

Common Error Types

Syntax error

- The code has not been correctly typed, a "typo" in the code.
- For example entering `print = (Hello` Instead of `Print = ("Hello")`

Logic error

- The code has been typed, there is an error in the logic used to create it.
- This might be running steps in the correct order, or multiplying instead of dividing

Key Terms

- Algorithmic thinking - identifying the steps involved in solving a problem.
- Algorithm – a series of steps to perform an action or solve a problem.
- Flowchart – a diagram showing inputs, outputs and processes within an algorithm.
- Process – an action that takes place.
- Pseudocode – simplified language used to design algorithms.
- Exam Reference Language – a more formal way of writing algorithms used within the exam.

Sequencing

- Breaking down complex tasks into simple steps.
- The order of steps matter
- Step by step progress through a program
- Benefits
  - Each line follows the next.
  - Can create simple programs very quickly.
  - Easy to follow for a small program.
- Disadvantages
  - Not very efficient.
  - Difficult to follow with large programs.
  - Hard to maintain.

Data Types

- Integers – whole numbers e.g. 27
- Reals – numbers containing decimals e.g. 56.2
- Boolean – TRUE or FALSE
- Strings – alphanumeric characters e.g. hello
- Casting is used to convert data from one type to another. This is often used to convert string input to integer or real to allow for calculation

Sub Programs

- Used to save time and simplify code
- Allows the same code to be used several times without having to write it out each time
- Procedures are sets of instructions stored under a single name (identifier)
- Functions are similar to procedures but will always return a value to the main program
- Parameters are values passed into a sub program. These are referred to as arguments when calling the sub program
- Both procedures and functions can accept parameters

Arrays

- An ordered collection of related data
- Each element in the array has a unique index, usually starting at 0
- All elements must be the same type of data
- Arrays are usually a fixed size
- 1D arrays are similar to a simple list, each element needs a single index number
- 2D arrays are similar to tables, with each element needing two index numbers
- 2D arrays are usually used to store properties of objects, with objects in rows and properties in columns
- Fruits[1] references element 1 in the 1D Fruits array
- Tools[0,2] references element 0,2 in the Tools array

The Use Of Records To Store Data And SQL

- Data is often stored in databases, providing persistent storage for data.
- Data within databases is stored in records, which in turn are stored in files.
- Records contain several attributes, each attribute is a single point of data.
- SQL (Structured Query Language) is a programming language designed for interacting with databases.
- SQL uses the SELECT command to search and read databases

SELECT * FROM Books	Returns all columns and records in the Books table
SELECT Title FROM Books	Returns only the title column from the Books table
SELECT * FROM Books WHERE Author="Bob"	Searches the Books table for records where the Author is Bob. Returns all Columns
SELECT * FROM Books WHERE Author="Bob" or Author="Tim"	Searches the Books table for records where the Author is Bob or Tim. Returns all Columns
SELECT * FROM Books WHERE Author!="Bob"	Searches the Books table for records where the Author is not Bob. Returns all Columns
SELECT Title FROM Books WHERE Sales>=100	Searches the Books table for records where Sales is greater than or equal to 100. Returns only the Title column

2.2 Programming Fundamentals

String Manipulation

- stringname.length – returns the length of a string
- stringname.upper – converts the string to uppercase

`string = "John"`		
`string.length`	The length of the string	4
`string.upper`	Convers to upper case	JOHN
`string.lower`	Converts to lower case	john
`string.substring(1,2)`	Returns part of the string	oh
`string.left(3)`	Returns from the left of the string	Joh
`string.right(2)`	Returns from the right hand side of the string	hn
`string+string`	Concatenates or joins strings	JohnJohn

Keywords

Variables:
- A box in which data may be stored
- Content changes as the program runs.
- Different types e.g. string, decimal, etc.

Assignment:
- The process for changing the data stored in a variable
- Copies a value into a memory location
- Different values may be assigned to a variable at different times during the execution of a program.
- Each assignment overwrites the current value with a new one.

Constants:
- Data does not change as the program runs
- Used to reference known values such as pi

Inputs:
- May come from the user, a file or elsewhere in a modular program
- Usually treated as text even if containing numbers

Outputs:
- The end result of the program
- May be displayed on the screen, written to a file, or sent to a device

Operators:
- Used to manipulate and compare data

Operators

Arithmetic Operators

+	Addition
-	Subtraction
*	Multiplication
/	Division
MOD	Modulus (the remainder from a division, e.g. 12 MOD 5 gives 2)
DIV	Quotient (integer division, e.g. 21 DIV 5 gives 4)
^	Exponentiation (to the power of, e.g. 3^3 gives 27)

Comparison Operators

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

Boolean Operators

AND - two conditions must be met for the statement to be true

OR - at least one condition must be met for the statement to be true

NOT – inverts the result, e.g. NOT(A AND B) will only be false when both A and B are true

File Handling Operations

- Files can be opened for reading or writing
- Append mode adds to the end of the file
- Write mode overwrites existing content in the file

`myFile := OPEN ("test.txt") FOR READING`	Opens test.txt in read mode into the myFile variable
`WHILE NOT myFile.EOF OUTPUT myFile.READLINE() END WHILE`	Uses a while loop to output each line of the file (READLINE) until the end of file (EOF) is reached.
`myFile.CLOSE()`	Closes the file

`myFile=OPEN ("logfile.txt") FOR APPEND`	Opens the logfile.txt file in append mode, meaning the existing content is preserved
`myFile.WRITELINE("This is a log entry")`	Writes to the end of the file
`myFile.CLOSE()`	Closes the file

`myFile=OPEN ("textfile.txt") FOR APPEND`	Opens the textfile.txt file in write mode, meaning the existing content will be overwritten
`myFile.WRITELINE("This is a log entry")`	Writes content to the file
`myFile.CLOSE()`	Closes the file

Selection

- Allows the program to make decisions
- Uses conditions to change the flow of the program
- Selections may be nested one inside another
- IF statements perform comparisons sequentially and so the order is important
- SELECT CASE has less typing but is less flexible

```
IF X > 50 THEN
      OUTPUT "A*"
ELSE IF X > 30 THEN
  OUTPUT "A"
ELSE
      OUTPUT "Fail"
END IF
```

```
SELECT CASE X
      CASE >100
            OUTPUT "A*"
      CASE >80
            OUTPUT "A"
      CASE >60
            OUTPUT "B"
      CASE ELSE
            OUTPUT "Fail"
END SELECT
```

Iteration

- Running through or 'iterating' through a set of steps several times.
- Also known as looping
- Count controlled iteration
  - Repeats the same code a set number of times
  - Uses a variable to track how many times the code has been run
  - This variable can be used in the loop
  - At the end of each iteration the variable is checked to determine if the code should be run again
  - FOR sets how many times the code should be repeated
  - NEXT tells the code to return to the start of the loop
  - STEP sets how the variable should increment
- Condition Controlled Iteration
  - Uses a condition to determine how many times code should be repeated
  - While loops will run whilst a condition is met and use the statements WHILE and ENDWHILE
  - Repeat loops will run until a condition is met and use the statements REPEAT and UNTIL

```
FOR count = 2 to 10 STEP 2
      OUTPUT count * 3
NEXT count

count = 0
WHILE count < 6
     print("Hello World")
     count = count + 1
ENDWHILE
```

Random Numbers

- Many different applications in computer programs from simulating dice in computer games, to cryptography
- Depending on the language we may specify just the maximum number assuming starting from 1 (e.g. roll = random(5)) or the first and last possible values (e.g. roll = (3,9))
- In many cases our desired output may not be a number and so we must then use selection, such as an IF or CASE statement, to convert the number into an actual choice
- We can also use the random number to select a random element from an array. This is more efficient then writing lots of IF statements.

Authentication

- A coding method to check that:
  - The user is who they say they are
  - The user is allowed to access the program
- Can be as simple as asking for a username and password
- There are three main authentication factors
  - Something you are, such as a fingerprint or iris scan.
  - Something you know, such as a password, pin or secret answer to a question.
  - Something you have such as a swipe card or mobile phone app.
- 2 factor authentication is where two different authentication types are required to access the program.

Logic Errors

- An error in the way the program works, causing it to not do what it should.
- May be the incorrect use of operators such as entering < instead of >
- May be the creation of an infinite loop.
- May be the accidental reuse of a variable name
- A program will run with logic errors but will not function correctly.

Syntax Errors

- A mistake in how the code is written, breaking the rules of the programming language.
- May be a misspelling or typo such as prnit instead of print.
- May be a missed bracket.
- May be using a variable without declaring it.
- A program will not run if there are syntax errors.

Input Validation

- Any user inputs may be incorrect, the program be able to handle this.
- Validation applies rules to inputs, data which does not follow the rules is rejected to prevent it from crashing the program.
  - Range Check – the input must be within a range. Usually applied to numbers and dates. For example, when inputting the required quantity into an order form, the number must be greater than 0 and less than the total stock available.
  - Length Check - the input must not be too long or too short. For example, a password must be at least eight characters, but not more than 30.
  - Presence Check – the input must be present. For example, requiring a credit card number for an online order
  - Format Check - the data must be in a specific format. For example, an email address must have an @ symbol and at least one dot.
  - Type Check - the data must be a specific type, such as requiring a currency input to be only numbers.
- Validation will not catch all errors as users may still make typos.
- Verification requires the user to enter key info twice to reduce the risk of this.

2.3 – Producing Robust Programs

Good Practice

```
VAR Password as String
VAR User as String
Password=Input("Enter the password")
#Ensure the password is correct
IF Password="letmein" THEN
      #Apply access levels
      IF User="Technician" THEN
            Allow Unrestricted Access
      ELSE
            Allow Restricted Access
ELSE
      Deny Access
END IF
```

Bad Practice

```
VAR X as String
VAR Y as String
X=Input("Enter the Password")
IF X="letmein" THEN
IF Y="Technician" THEN
Allow Unrestricted Access
ELSE
Allow Restricted Access
ELSE
Deny Access
END IF
```

Naming Conventions

- Using the same rules for naming throughout the program make it easier to read.
- These are applied to variables, functions, procedures, etc.
- Should be easy to read.
- Should be meaningful.
- Makes the code easier to read and to understand.
- For Example, FirstName is a better variable name than just X or FN.

Test Plans

- Provides structure to testing.
- Records the result of testing.
- Should include:
  - The test number
  - The data entered
  - The type of data
  - The expected outcome
  - The result of the test
  - Any action required as a result

Refining Algorithms

- User prompts should be helpful and explain any input validation rules.
- Code should convert inputs to the required data type if needed.
- Loops can be used to request the user re-enter data if it is invalid.
- There may be a limit on how many times the user is asked in the case of passwords or other security fields.

Selecting and Using Suitable Test Data

- A range of data should be used when testing.
- Normal data is correct and what would usually be inputted by the user.
- Boundary data is correct but is the largest or smallest value which a user might input. For example, entering an age of 105.
- Invalid data is too large or small, for example entering an age of 2978.
- Erroneous data is completely incorrect, for example entering Bob into an age field.

Indentation

- Allows code within a particular function or procedure to be grouped together.
- Often used with IF statements.
- Multiple levels of indentation may be used.
- Makes the code easier to read and understand.
- Makes it easier to focus on particular parts of the code when needed.

Commenting

- Lines within the code which are not executed.
- Starts with a certain character depending on the language used. Common symbols include # */ and /
- Informs the reader about bugs or issues in the code
- Explains the functionality of particular code
- Explains the purpose of particular code
- Prevent code from executing without deleting it completely.

Exam Style Question

Explain, using examples, way to improve the maintainability of the program shown above [4]

Indent the lines within the IF statement in order to make the code easier to read.
Use sensible variable names, for example 'X' could instead be called 'Password' and 'Y' could be called 'User'. This would make the program easier to read.

Anticipating Misuse

- May be a brute force attack on the program.
- May be a user entering an incorrect input to try and break the program.
- May be a user entering code into input fields to access parts of the program they should not.
- May simply be an error in input.

Sub Programs

- Procedures carry out a set of instructions and do no not return a value.
- Functions are similar but do will return a value.
- Both procedures and functions can accept parameters
- Parameters are values passed into a sub program. These are referred to as arguments when calling the sub program
- They provide structure to the code.
- They make code easier to understand.
- They allow code to be easily reused.
- They allow the program to be shorter as code need not be written out multiple times.

Testing

- Newly written code often contains errors.
- Testing helps to locate and remove these errors.
- Testing ensures the program works in the way it should.

Iterative Testing

- Takes place whilst the program is being written.
- The programmer tests individual lines or sections of code as they are written.
- If an error is found, it is fixed and the code tested again.
- This process repeats, or iterates, until the code works as intended.
- It is easier to fix errors in smaller sections of code.

Final Testing

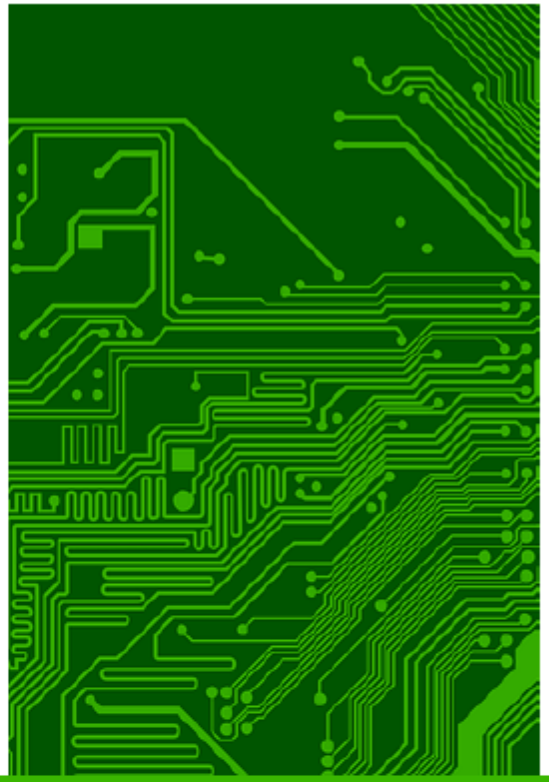
- Takes place once the code is finished.
- A final check to make sure the code works correctly.
- Makes sure the program does what it should.
- It can be harder to locate and fix errors at this stage because of the amount of code.

Some Background

- A computer's CPU is made up of millions of tiny switches called transistors.
- These switches can be either on or off.
- We therefore use binary to represent these switches, since a binary digit can be either 0 or 1.
- 0 represents a transistor which is off, 1 represents one which is on.

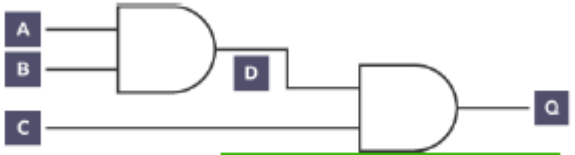
Truth Tables

- Used to show the output of logic gates or logic circuits.
- To create a truth table:
  - Calculate how may rows are needed (2number of inputs)
  - So 4 inputs would need 24 or 16 rows
  - List the values for each input
  - Work through the diagram to complete the output for each possible input



Bringing It All Together

- Two or more logic gates are often used one after the other.
- This could be several of the same gate, or several different gates.
- This is known as a Logic Circuit.
- It is important to consider the order in which the gates are used.
- We can use diagrams and truth tables to represent these as shown below.



A	B	C	D	Q
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	0
1	0	0	0	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

2.4 – Boolean logic

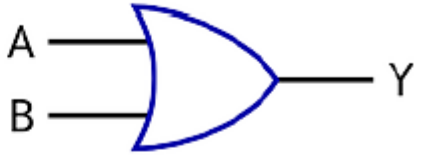
The AND Gate

- Will output 1 if both A and B are 1.
- Will output 0 if either A or B is 0.
- Written as A ∧ B

A	B	A ∧ B
0	0	0
0	1	0
1	0	0
1	1	1

The OR Gate

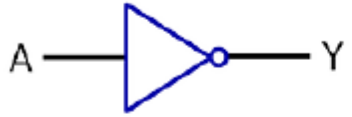
- Will output 1 if either A or B are 1
- Will output 0 if both A and B are 0
- Written as A∨B



A	B	A ∨ B
0	0	0
0	1	1
1	0	1
1	1	1

The NOT Gate

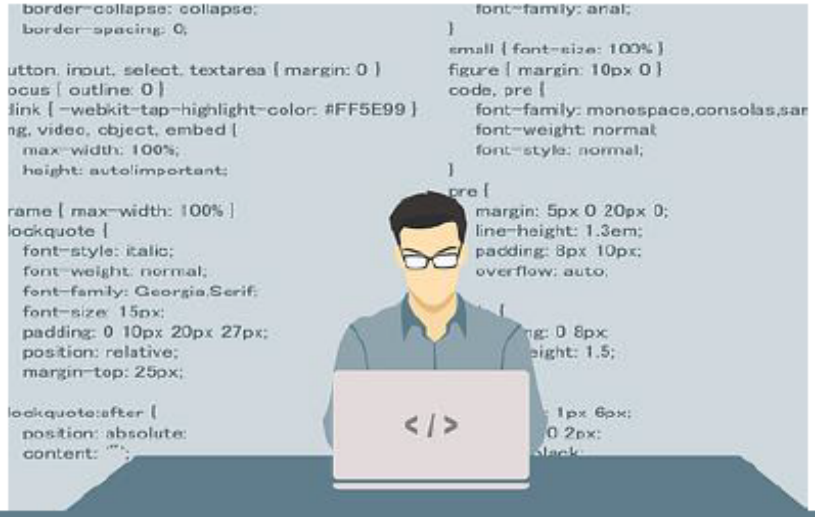
- Has a single input
- Inverts the input (1 becomes 0 and 0 becomes 1)
- Written as NOT A



A	NOT A
0	1
1	0

Key Terms

- **Logic Gate** – components which compare one or more inputs based on a logical function to provide a single output.
- **Logic Diagram** – a diagram showing one or more logic gates.
- **Transistor** – components contained in the CPU which can be either on or off.
- **Truth Table –** a table representing the possible outputs of a logic gate or diagram
- **Logic Circuit –** two or more logic gates used together one after the other
- **Binary** – a number system containing two symbols, 0 and 1. Also known as Base 2



IDEs (Integrated Development Environments)

A software package containing several features useful for writing code:

Editor
- Allows code to be to written and edited
- Fairly simple with programmer specific features
- Automatic line numbering
- Colour coding
- Auto-correct
- Auto-suggestion
- Auto-indent

Error diagnostics and debuggers
- Help to locate and fix errors.
- Breakpoints allow a program to be paused at certain points so the programmer can then examine different parts of the code or variables.
- Variable tracing shows the changing values of variables as the program runs
- Syntax highlighting shows where syntax errors occur.

Runtime environment
- Software which allows code to run on different platforms to that which they were written on.
- Allows code to be written for specialist hardware without having to have that hardware to hand.
- Creates a virtual machine to run the code in

Translators
- Translates code into a format which the computer can execute.
- Allows code to be run and tested from within the IDE.

2.5 – Programming Languages and Integrated Development Environments

Low Level Languages

1st Generation

- Also known as Machine Code
- Can be executed directly by the CPU
- The generation that 'computers understand'
- Difficult for humans to understand, write or debug

```
01010101010100101010100
10101010101001111100100
0100001010100101
```

2nd Generation

- Also known as Assembly Code
- Uses mnemonics (abbreviations)
- Easier for humans to understand and program but still difficult
- 1-1 relationship with Machine Code (one Assembly Language instruction translates to one Machine Code Instruction
- Must be translated into Machine Code for execution
- Commonly used to program device drivers

```
LOAD r1, c
LOAD r2, d
ADD r1, r2
DIV r1, #2
```

High Level Languages

3rd Generation

- Easier still for humans to understand, program and debug
- Uses English-Like Keywords
- 1-many relationship with machine code (one instruction translates into many machine code instructions)
- Examples include Java, Basic and Pascal
- Translated using a compiler or interpreter

```
Dim Num1, Num2, Tot as Integer
Num1 = Console.Readline()
Num2 = Console.Readline()
Tot = Num1 + Num2
```

4th Generation

- Also known as Declarative Languages
- Has strict facts and rules for use
- Details the computation which should be performed but not how to perform it
- Examples include SQL, Expert Systems and Artificial Intelligence

```
SELECT * FROM Customers
WHERE Country='Mexico';
```

Translators

- Translates a language into a form that the computer can directly execute
- Compilers and interpreters are used for 3rd generation languages

Compilers

- Translates the whole code in one go into Machine Code.
- Optimise the code
- Used at the end of development when code is finished
- Create error reports and object code

Interpreters

- Translate and execute source code
- Work line by line.
- Syntax is checked
- If code is correct it is executed
- If code is incorrect interpreting is stopped.
- Aid debugging

	Compilers	Interpreters
Advantages	• Compiled programs run quickly and without needing additional software. • Compiled programs can be supplied as an executable file which cannot be easily modified. • Optimise code so it runs quickly and uses less memory.	• Instructions are executed as soon as they are translated. • Instructions are not stored for later so less memory is needed. • Errors can be quickly spotted.
Disadvantages	• Because the source code is translated as a whole, more memory is needed. • Requires a temporary working space for the compiler to perform the translation. • Do not spot errors. • Code must be re-compiled every time it changes. • Code compiled on one platform will not run on another.	• The CPU must wait for each instruction to be translated so execution is slower. • Code is translated each time it is run. • Do not produce an executable file that can be distributed • Do not optimise code.

Knowitall - Computer Systems Unit 1		Module	Grade %	Note
1	Units & Numbers	Units & Data Storage		
2	Binary & Decimal Numbers			
3	Binary Addition			
4	Hexadecimal Numbers			
5	Binary Shifting			
6	CPU Architecture	Systems Architecture		
7	Common CPU Components			
8	The Fetch-Execute Cycle			
9	CPU Performance & Embedded Systems			
10	Primary Storage	Primary & Secondary Storage		
11	Secondary Storage 1			
12	Secondary Storage 2			
13	Character Sets & Images	Data Representation & Compression		
14	Sound			
15	Compression			
16	Types of Networks	Networks & Topologies		
17	Network Hardware			
18	The Internet			
19	Network Topologies			
20	Modes of Connection	Wired & Wireless Networks, Protocols & Layers		
21	Encryption			
22	Addresses & Standards			
23	Common Protocols & Layers			
24	Forms of Attack 1	Network Security		
25	Forms of Attack 2			
26	Identifying & Preventing Vulnerabilities 1			
27	Identifying & Preventing Vulnerabilities 2			
28	Operating Systems 1	Systems Software		
29	Operating Systems 2			
30	Utility Software			
31	Ethical & Cultural Issues	Ethical, Legal, Cultural & Environmental Impact		
32	Environmental & Privacy Issues			
33	Data Protection & Computer Misuse			
34	Copyright & Licencing			

	Principles of Computer Science U2	Module	Grade %	Note
1	Decomposition	Computational Thinking		
2	Pattern Recognition, Generalisation & Abstraction			
3	Algorithm Design			
4	Interpreting Pseudocode	Structured English & Flowcharts		
5	Developing Pseudocode			
6	Flowcharts Using Standard Symbols			
7	Constants & Variables	Handling Data within a Program		
8	Arithmetic Operations			
9	Arithmetic Functions	Built-In Functions		
10	String Functions			
11	String Conversions			
12	General Functions			
13	Validating Data 1	Validating Data		
14	Validating Data 2			
15	Error Handling and Reporting			
16	Loops	Control Structures		
17	Branches			
18	Function Calls			
19	Data Structures & Strings	Data Structures		
20	One-Dimensional Arrays			
21	Two-Dimensional Arrays			
22	Records			
23	Linked Lists			
24	Sets			
25	Stacks	Standard Algorithms – Data Structures		
26	Queues			
27	Bubble Sort	Standard Algorithms – Sorting		
28	Insertion Sort			
29	Quicksort and Recursion			
30	Quicksort			
31	Linear Search	Standard Algorithms – Searching		
32	Binary Search			
33	Count Occurrence			
34	Procedural Programming	Types of Programming Languages		
35	Object-Oriented Programming 1			
36	Object-Oriented Programming 2			
37	Object-Oriented Programming 3			
38	Event-Driven Programming			
39	Mark-Up & Web Languages	Coding for the Web		
40	Client-Side Scripting			
41	Server-Side Scripting			
42	Language Translation	Translation		

	Knowitall - **Computational Thinking, Algorithms & Programming**	Module	Grade%	Note
1	Logic Diagrams	Boolean Logic		
2	Truth Tables			
3	Combining Gates & Solving Problems			
4	Computational Thinking	Computational Thinking		
5	Problem Structure	Algorithm Design		
6	Pseudocode & Natural English			
7	OCR Exam Reference Language			
8	Flowcharts			
9	Improving Algorithms			
10	The Use of Outputs	Programming Fundamentals & Data Types		
11	Variables & Data Types			
12	Variables, Constants & Input			
13	Casting Variables			
14	The Common Operators			
15	Sequence and Selection 1	Sequence, Selection & Iteration		
16	Sequence and Selection 2			
17	Sequence and Selection 3			
18	Iteration 1			
19	Iteration 2			
20	Basic String Manipulation 1	String Manipulation & File Handling		
21	Basic String Manipulation 2			
22	Basic File Handling Operations			
23	One-Dimensional Arrays	Data Structures		
24	Two-Dimensional Arrays			
25	Records			
26	SQL			
27	Procedures	Sub Programs		
28	Functions			
29	Defensive Design Considerations	Producing Robust Programs		
30	Input Validation			
31	Maintainability			
32	Testing			
33	Language Types & Translators	Programming Languages & IDEs		
34	The Integrated Development Environment			
35	Linear Search	Searching & Sorting Algorithms		
36	Binary Search			
37	Bubble Sort			
38	Insertion Sort			
39	Merge Sort			

Code Academy			Name	
Learn Python 2 Progress Sheet.				
	Section	Subsections	Date Completed	What you learned ? Just a quick short note of a piece of useful code.
1	Python Syntax	Python Syntax		
2	Strings and Console Output	Strings and Console Output		
		Date and Time		
3	Conditionals and Control Flow	Conditionals and Control Flow		
		PygLatin		
4	Functions	Functions		
		Taking a Vacation		
5	Lists & Dictionaries	Python Lists & Dictionaries		
6	Student Becomes the Teacher	Student Becomes the Teacher		
7	Lists and Functions	Lists and Functions		
		Battleship		
8	Loops	Loops		
		Practice makes perfect		
9	Exam Statistics	Exam Statistics		
10	Advanced Topics in Python	Advanced Topics in Python		
11	Introduction to Classes	Introduction to Classes		
		classes		
12	File Input and Output	File Input and Output		

Edulito - End of unit tests			Target grade:
	Max	Actual	Review
1.1 Systems Architecture	30		
1.2 test 1 Memory	28		
1.2 test 2 Storage	28		
1.2 test 3 Data storage	91		
1.3 Test 1 Networks Protocols	50		
1.3 Test 2 Wire edless protocols Layers	41		
1.4 Network Security	32		
1.5 Systems software	26		
1.6 Ethical Cultural Environment	31		
2.1 Algorithms	63		
2.2 Programming fundamentals	71		
2.3 Robust Programming	38		
2.4 Boolean Logic	46		
2.5 Languages and ideas.	35		

OCR Exam Reference Language

Logical operators

```
AND OR NOT
```

Example

```
while x<=5 AND flag==false
```

Comparison operators

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

Arithmetic operators

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

Comments

Concept	Keyword(s) / Symbols
Comment	//

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

Taking input from the user/keyboard

Concept	Keyword(s) / Symbols
Input	input(…)

variable=input(*prompt to user*)

Example

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

Outputting to the screen

Concept	Keyword(s) / Symbols
Output	print(…)

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

Example

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

Variables, constants and assignments

Concept	Keyword(s) / Symbols
Assignment	=
Constants	const
Global variables	global

Variables and constants are assigned using the = operator.

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

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

Variables and constants that are declared inside a function or procedure are local to that subroutine.
Variables in the main program can be made global with the keyword `global`.

```
global userid = 123
```

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

```
const vat = 20
```

Casting

Concept	Keyword(s) / Symbols
Converting to another data type	str() int() float() real() bool()

Variables can be typecast using the following functions.

```
str(3) returns "3"
int("3") returns 3
float("3.14") returns 3.14
real("4.52") returns 4.52
bool("True") returns TRUE
```

Iteration: count-controlled

FOR loop

Concept	Keyword(s) / Symbols
FOR loop	for… to … next …

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

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

FOR loop with step

Concept	Keyword(s) / Symbols
FOR loop	for… to … step… next …

```
for i = 2 to 10 step 2
   print(i)
next i
```

This will print the even numbers from 2 to 10 inclusive.

Iteration: condition-controlled

WHILE loop

Concept	Keyword(s) / Symbols
WHILE loop	while… endwhile…

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

DO WHILE loop

Concept	Keyword(s) / Symbols
DO WHILE loop	do… until…

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

Selection

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

IF-THEN-ELSE

Concept	Keyword(s) / Symbols

IF-THEN-ELSE	if… then
	elseif … then
	else
	endif

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

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

CASE SELECT or SWITCH

Concept	Keyword(s) / Symbols
CASE SELECT or SWITCH	switch…: case…: case…: default: endswitch

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

String handling

To get the length of a string:

Concept	Keyword(s) / Symbols
String length	.length

```
stringname.length
```

To get a substring:

Concept	Keyword(s) / Symbols
Substrings	.substring(x, i)

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

NB The string will start with the 0th character.

Converting cases:

Concept	Keyword(s) / Symbols
Uppercase	.upper
Lowercase	.lower

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

Ascii conversion:

Concept	Keyword(s) / Symbols
ASCII Conversion	ASC(…) CHR(…)

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

Example

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

Will display
16
Put

Subprograms

Concept	Keyword(s) / Symbols
Procedure	procedure name(…) endprocedure
Calling a procedure	procedure(parameters)
Function	function name(…) … return … endfunction
Calling a function	function(parameters)

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

Called from main program

```
y=triple(7)
   procedure greeting(name)
   print("hello"+name)
endprocedure
```

Called from main program

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

Arrays and lists

Concept	Keyword(s) / Symbols
Declaration	array names[…] array gameboard[…,…]
Assignment	names[…] = … gameboard[…,…] = …

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

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

Example of 2D array:

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

Reading to and writing from files

Concept	Keyword(s) / Symbols
Open	open(…)
Close	.close()
Read line	.readLine()
Write line	.writeLine(…)
End of file	.endOfFile()
Create a new file	newFile()

To open a file to read or write to it open is used. We then use writeLine to write a line to the file and readLine to return a line of text from the file.
The following program makes x the first line of sample.txt

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

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

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

In the program below, `hello world` is made the contents of sample.txt (any previous contents are overwritten).

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

To create a new file called "myNewFile.txt"

```
newFile = ("myNewFile.txt")
```

The file would then need to be opened using the above command for Open.

Random numbers

Concept	Keyword(s) / Symbols
Random number	Random(…,…)

```
myVariable = random(1,6)
```

Creates a random integer between 1 and 6 inclusive.

```
myVariable = random(-5.0,5.0)
```

Creates a random real between -5.0 and 5.0 inclusive.

No.	Spec	Section	Sub-topic	Term	Definition
1.	1.1.1	1.1 Systems architecture	Architecture of the CPU	CPU	**Central Processing Unit:** "The main part of the computer, consisting of the registers, ALU and control unit."
2.	1.1.1			Fetch-decode-execute cycle	"The complete process of retrieving an instruction from storage, decoding it and carrying it out. Also known as the instruction cycle."
3.	1.1.1			ALU	**Arithmetic Logic Unit:** "Performs calculations (e.g., x = 2 + 3) and logical comparisons (e.g., IF x > 3) in the CPU."
4.	1.1.1			CU	**Control Unit:** "Decodes instructions. Sends signals to control how data moves around the CPU."
5.	1.1.1			Cache	"Memory in the processor that provides fast access to frequently used instructions and data."
6.	1.1.1			Register	"Tiny areas of extremely fast memory located in the CPU, normally designed for a specific purpose where data or control information is stored temporarily – e.g., MAR, MDR, etc."
7.	1.1.1			Von Neumann architecture	"Traditional computer architecture that forms the basis of most digital computer systems. Instructions are fetched, decoded and executed one at a time."
8.	1.1.1			MAR	**Memory Address Register:** "Holds the address of data ready to be used by the memory data register or the address of an instruction passed from the program counter. Step two of the fetch-decode-execute cycle."
9.	1.1.1			MDR	**Memory Data Register:** "Holds data fetched from or to be written to memory. Step three of the fetch-decode-execute cycle."
10.	1.1.1			Program counter	"Holds the address of the next instruction to be executed. Step one of the fetch-decode-execute cycle."
11.	1.1.1			Accumulator	"Holds the result of calculations."
12.	1.1.2		CPU performance	Clock speed	"Measured in hertz, the clock speed is the frequency at which the internal clock generates pulses. The higher the clock rate, the faster the computer may work. The clock is the electronic unit that synchronises related components by generating pulses at a constant rate."
13.	1.1.2			Cache size	"The larger the cache, the more data that can be stored without having to go back to main memory (RAM) – this has a significant impact on processing speed."
14.	1.1.2			Cores	"Part of a multi-core processor, a single component with two or more independent CPUs that facilitate the fetch-decode-execute cycle."
15.	1.1.3		Embedded systems	Embedded system	"A computer built to solve a highly specific problem. Not easy to change. For example, the operating system placed inside a washing machine, microwave or set of traffic lights."
16.	1.2.1	1.2 Memory and storage	Primary storage (Memory)	Primary storage	"Comprised of random-access memory (RAM) and read-only memory (ROM). It holds data and instructions that the CPU can access more quickly and easily than from secondary storage devices."
17.	1.2.1			RAM	**Random-Access Memory:** "Volatile (data is lost when the computer is powered off). Read-and-write. Purpose: Temporary storage of currently executing instructions and data – e.g., applications and the operating system."
18.	1.2.1			ROM	**Read-Only Memory:** "Non-volatile (data is retained when the computer is powered off). Read-only. Purpose: Stores startup instructions, otherwise known as the bootstrap."
19.	1.2.1			Virtual memory	"Using part of the hard disk as if it were random-access memory. Allows more applications to be open than physical memory can hold."
20.	1.2.2		Secondary storage	Secondary storage	"Permanent storage of instructions and data not currently in use by the processor. Stores the operating system, applications and data. Read-and-write and non-volatile."
21.	1.2.2			Optical storage	"CD-R, CD-RW, DVD-R, DVD-RW. Use: Music, films and archive files. Low capacity. Slow access speed. High portability. Prone to scratches. Low cost."
22.	1.2.2			Magnetic storage	"Hard disk drive. Use: Operating system and applications. High capacity. Medium data access speed. Low portability (except for portable drives). Reliable but not durable. Medium cost."
23.	1.2.2			Solid-state storage	"Memory cards and solid-state hard drives (SSD). Use: Digital cameras and smartphones. Medium capacity. High portability. Reliable and durable. No moving parts. Fast data access speed. High cost."
24.	1.2.2			Storage capacity	"The amount of data a storage device can store."
25.	1.2.2			Storage speed	"The read/write access speed of a storage device."
26.	1.2.2			Storage portability	"How easy it is to transport a storage device – e.g., solid-state and optical storage are highly portable, whereas magnetic storage is designed to stay in place."
27.	1.2.2			Storage durability	"How resistant a storage device is to damage and wear. Devices with low durability are likely to fail earlier."
28.	1.2.2			Storage reliability	"A relative measure of confidence that a storage device will function correctly and allow you to write, read, delete and modify data."
29.	1.2.2			Storage cost	"The relative price of a storage device – e.g., per megabyte of data."
30.	1.2.3		Units	Bit	"The smallest unit of storage, represented by either a binary 1 or 0."
31.	1.2.3			Nibble	"Half a byte. Four bits."
32.	1.2.3			Byte	"A collection of eight bits."
33.	1.2.3			Kilobyte	"One kilobyte (KB) is 1024 bytes. For the purpose of calculations in an exam, you can treat a kilobyte as 1000 bytes."
34.	1.2.3			Megabyte	"One megabyte (MB) is 1024 kilobytes (KB). For the purpose of calculations in an exam, you can treat a megabyte as 1000 KB."
35.	1.2.3			Gigabyte	"One gigabyte (GB) is 1024 megabytes (MB). For the purpose of calculations in an exam, you can treat a gigabyte as 1000 MB."
36.	1.2.3			Terabyte	"One terabyte (TB) is 1024 gigabytes (GB). For the purpose of calculations in an exam, you can treat a terabyte as 1000 GB."
37.	1.2.3			Petabyte	"One petabyte (PB) is 1024 terabytes (TB). For the purpose of calculations in an exam, you can treat a petabyte as 1000 TB."

38.	1.2.4	1.2 Memory and storage	Data storage (Numbers)	Denary numbers	"A numerical system of notation that uses 10 as its base. The ten decimal base digits are 0 – 9."
39.	1.2.4			Binary numbers	"Binary describes a numbering scheme with only two possible values for each digit, 0 and 1. In computing, binary refers to any digital encoding system with exactly two possible states – e.g., in memory, storage, processing and communications, 0 and 1 are sometimes called low and high, respectively."
40.	1.2.4			Binary arithmetic	"The process of adding two or more positive 8-bit binary numbers (0 – 255)."
41.	1.2.4			Overflow	"The generation of a number that is too large to be represented by the device intended to store it."
42.	1.2.4			Hexadecimal	"A numerical system of notation that uses 16 rather than 10 as its base. The 16 hex base digits are 0 – 9 and the letters A – F."
43.	1.2.4			Binary shifts	"Allows you to easily multiply or divide a base-2 binary number. A left shift multiplies the number by 2, while a right shift divides it by 2.
44.	1.2.4		Data storage (Characters)	Character set	"A set of symbols represented by a computer. These symbols, called characters, can include letters, digits, spaces, punctuation marks and control characters."
45.	1.2.4			ASCII	**America Standard Code for Information Interchange:** "A character set devised for early telecommunication systems but proved to be ideal for computer systems. Uses 7 bits, providing 32 control codes and 96 displayable characters. The eighth bit is often used for error checking."
46.	1.2.4			Unicode	"Standard character set that replaces the use of multiple different character sets. Incorporates characters from almost all global languages. A 16-bit extension of ASCII."
47.	1.2.4		Data storage (Images)	Pixels	"The smallest unit of a digital image or graphic that can be displayed on a digital device. A pixel is represented by a dot or square on a computer display."
48.	1.2.4			Metadata	"A collection of data that describes and provides information about other data."
49.	1.2.4			Colour depth	"Also known as bit depth. Either the number of bits used to indicate a) the colour of a single pixel in a bitmap image or video frame buffer or b) each colour component of a single pixel."
50.	1.2.4			Resolution	"The number of pixels (individual points of colour) in a display, expressed in terms of the number of pixels on the horizontal and vertical axes."
51.	1.2.4			Image quality	"The overall detail of an image, affected by colour depth and resolution."
52.	1.2.4			Image file size	"The total size of an image file in storage. *Size in bits = Width in pixels * Height in pixels * Colour depth in bits*."
53.	1.2.4		Data storage (Sound)	Sample rate	"The number of samples taken per second, measured in hertz (Hz)."
54.	1.2.4			Sample duration	"How many seconds of audio a sound file contains."
55.	1.2.4			Sample bit depth	"The number of bits available to store each sample (e.g., 16-bit)."
56.	1.2.4			Playback quality	"The finished quality of the digital sound file – this is affected by the sample rate and bit depth. The higher the number, the better the quality and the larger the file size. CD quality is 44,100 samples per second."
57.	1.2.4			Sound file size	"The total size of a sound file in storage. *Size in bits = Sampling rate * Sample resolution * Number of seconds.*"
58.	1.2.5		Compression	Compression	"The process of reducing the size of a file."
59.	1.2.5			Lossy compression	"A compression method that generally involves a loss of quality where experience tells us that it will be least noticed."
60.	1.2.5			Lossless compression	"A compression method that allows a file to be recreated in its original quality."

61.	1.3.1	1.3 Computer networks, connections and protocols	Networks and topologies	LAN	**Local Area Network:** "Small geographic area. All hardware is owned by the organisation using it. Wired with UTP or fibre optic cable or wireless using routers and Wi-Fi access points."
62.	1.3.1			WAN	**Wide Area Network:** "Large geographic area. Infrastructure is hired from telecommunication companies who own and manage it. Connected with telephone lines, fibre optic cables or satellite links."
63.	1.3.1			Client-server network	"A client makes requests to the server for data and connections. A server controls access and security to one shared file store. A server manages access to the internet, shared printers and email services, as well as running data backups."
64.	1.3.1			Peer-to-peer network	"All computers are equal and serve their own files to each other. Each computer is responsible for its own security and backups and usually has its own printer."
65.	1.3.1			Wireless access point	"Hardware that allows a Wi-Fi-enabled device to connect to a network."
66.	1.3.1			Router	"A router sends data between networks. It is needed to connect a local area network to a wide area network. It uses the IP address on a device to route traffic to other routers."
67.	1.3.1			Switch	"A switch sends data between computers on a local area network. It uses the NIC address on a device to route traffic."
68.	1.3.1			NIC	**Network Interface Card/Controller:** "Hardware that connects a computer to a network."
69.	1.3.1			Transmission media	"Physical media that can be used to transmit data – e.g., twisted copper cable, fibre optic, etc."
70.	1.3.1			The internet	"A worldwide collection of interconnected computer networks. An example of a WAN – the largest in existence."
71.	1.3.1			DNS	**Domain Name System:** "The internet equivalent of the phone book. Maintains a directory of domain names and translates them to Internet Protocol (IP) addresses – this is necessary because, although domain names are easy to remember, computers access websites using IP addresses."
72.	1.3.1			Hosting	"Websites stored on dedicated servers. Used for websites that need to be available 24/7, be accessed by thousands of users at a time, be well-protected from hackers and have an IP address that doesn't change."
73.	1.3.1			The cloud	"Remote servers that store data to be accessed over the internet. Access anytime, anywhere from any device. Automatic backups. Collaborate on files easily."
74.	1.3.1			Web server	"A program that uses HTTP (Hypertext Transfer Protocol) to deliver web pages to users. Page requests are forwarded by a computer's HTTP client. Dedicated computers and appliances may also be referred to as web servers."
75.	1.3.1			Client	"A device that requests and/or uses services from a remote/connected server."
76.	1.3.1			Network topology	"The physical or logical arrangement of connected devices on a network – e.g., computers, switches, routers, printers, servers, etc."
77.	1.3.1			Star topology	"Computers connected to a central switch. If one computer fails, no others are affected. If the switch fails, all connections are affected."
78.	1.3.1			Mesh topology	"Switches/routers connected so there is more than one route to the destination – e.g., the internet. More resilient to faults but more cable is required."
79.	1.3.2			Wired connection	"Any computer network that predominantly connects hardware via physical cables – e.g., copper, fibre optic, etc."
80.	1.3.2			Ethernet	"A standard for networking local area networks using protocols. Frames are used to transmit data. A frame contains the source and destination addresses, the data and error-checking bits. Uses twisted pair and fibre optic cables. A switch is used to connect computers."
81.	1.3.2			Wireless connection	"Any computer network that predominantly connects hardware via Wi-Fi, eliminating much of the need for physical cabling."
82.	1.3.2			Wi-Fi	"Wireless connection to a network. Requires a wireless access point or router. Data is sent on a specific frequency. Each frequency is called a channel."
83.	1.3.2			Bluetooth	"A method of exchanging data wirelessly over short distances – much shorter than Wi-Fi. Examples of typical Bluetooth use could be, headphones, car mobiles etc."
84.	1.3.2			Encryption	"Encoding readable data (plain text) into unreadable data (ciphertext). Only the intended recipient can decode the data using a special key. Protects sensitive communications against hacking."
85.	1.3.2			IP address	**Internet Protocol Address:** "A unique string of numbers separated by full stops. Identifies each computer using IP to communicate via a network."
86.	1.3.2			MAC address	**Media Access Control Address:** "Used as a unique identifier for most network technologies including Ethernet and Wi-Fi."
87.	1.3.2			Standards	"Various rules for different areas of computing. Standards allow hardware and software from different manufacturers to interact with each other."
88.	1.3.2			Protocol	"A set of rules that allow two devices to communicate."
89.	1.3.2			TCP/IP	**Transmission Control Protocol/Internet Protocol:** "TCP provides error-free transmission between two routers. IP routes packets across a wide area network."
90.	1.3.2			HTTP	**Hypertext Transfer Protocol:** "A client-server method of requesting and delivering HTML web pages. Used when the information on a web page is not sensitive or personal."
91.	1.3.2			HTTPS	**Hypertext Transfer Protocol Secure:** "Encryption and authentication for requesting and delivering HTML web pages. Used in websites that are sending and/or receiving sensitive data (e.g., passwords, bank details)."
92.	1.3.2			FTP	**File Transfer Protocol:** "Used for sending files between computers, usually on a wide area network."
93.	1.3.2			POP	**Post Office Protocol:** "Used by email clients to retrieve email from an email server."
94.	1.3.2			IMAP	**Internet Message Access Protocol:** "Used by mail clients to manage remote mailboxes and retrieve email from a mail server."
95.	1.3.2			SMTP	**Simple Mail Transfer Protocol:** "Sends email to a mail server."
96.	1.3.2			Protocol layering	"The concept of protocol rules being built up in layers – the layered protocol stack facilitates the various rules being executed in a defined order."

97.	1.4.1	1.4 Network security	Threats to computer systems and networks	Malware	“A broad term that covers all software written to facilitate loss of data, encryption of data, fraud and identity theft.”
98.	1.4.1			Social engineering	“Most vulnerabilities are caused by humans – not locking computers, using unsecure passwords, not following company network policy or implementing it poorly, not installing protection software, not being vigilant with suspicious emails/files and not encrypting sensitive data.”
99.	1.4.1			Phishing	“Sending emails purporting to be from reputable companies to entice people into revealing personal information.”
100.	1.4.1			Brute-force attack	“A trial-and-error method of attempting to guess passwords. Automated software is used to generate a large number of guesses.”
101.	1.4.1			Denial-of-service attack	“Flooding a server with so much traffic that it cannot process legitimate requests.”
102.	1.4.1			Data interception and theft	“Stealing computer-based information.”
103.	1.4.1			SQL injection	“A hacking technique used to view or change data in a database by inserting SQL code into a form instead of data.”
104.	1.4.2		Identifying and preventing vulnerabilities	Penetration testing	“Designed to test the security of a system and identify vulnerabilities.”
105.	1.4.2			Anti-malware software	“Protects against many types of malware including viruses, worms, trojans, rootkits, spyware, key loggers, ransomware and adware.”
106.	1.4.2			Firewall	“Network software or hardware designed to prevent external users from gaining unauthorised access to a computer system.”
107.	1.4.2			User access level	“The degree of system access that a specific type of user is allowed. On a network, most users will have restricted access, whereas a system administrator or network technician will be allowed much greater access with fewer restrictions.”
108.	1.4.2			Password	“A secret word or phrase used to gain access to a computer, program, interface or system.”
109.	1.4.2			Physical security	“Any form of physical security intended to protect data and systems – e.g., alarms, locks, security patrols, etc.”
110.	1.5.1	1.5 System software	Operating systems	System software	“Software that manages the computer. Usually supplied with the computer.”
111.	1.5.1			Operating system	“Specialised software that communicates with computer hardware to allow other programs to run. The most common operating systems are Windows, Linux, Unix, MacOS and iOS.”
112.	1.5.1			User interface	“Allows a user to interact with a computer – e.g., input devices and software.”
113.	1.5.1			Memory management	“The process of the operating system deciding what should be in memory at any given time. Responsible for loading data and programs into and out of memory when required.”
114.	1.5.1			Multitasking	“Running multiple applications simultaneously by giving each one a slice of processor time.”
115.	1.5.1			Peripheral management	“The management of connected input/output devices such as a mouse, keyboard, webcam, speaker, scanner, printer, etc.”
116.	1.5.1			Driver	“Translates operating system commands into hardware-specific commands – e.g., a printer driver tells the printer how to print a document.”
117.	1.5.1			User management	“Allows different people to log into the same computer with a username and password. Remembers personal settings. Manages file access rights.”
118.	1.5.1			File management	“Access permissions for files (read and write). Opening files in programs. Moving, deleting and renaming files. Presenting a folder structure to the user.”
119.	1.5.2		Utility software	Utility software	“A program that performs a specific task relating to the operation of the computer – e.g., backup, virus scan, compression, defragmentation.”
120.	1.5.2			Encryption software	“Turns plaintext data into unreadable ciphertext data using a key. Protects data from being read by hackers.”
121.	1.5.2			Defragmentation software	“Files being deleted over time creates gaps on a hard disk. New files fill the gaps but may need more space than the gap provides, resulting in file fragments being spread across the disk. Defragmentation puts file fragments and free space back together in contiguous space, improving access speeds.”
122.	1.5.2			Data compression software	“Reduces the size of a file so it takes up less disk space and is quicker to download over the internet. Compressed files must be extracted before they can be read.”

123.	1.6.1	1.6 Ethical, legal, cultural and environmental concerns	Ethical, legal, cultural and environmental impact	Ethical issues	“Ethical issues introduced by the increasing use of computer science and its related technologies – e.g., job losses, AI/machine learning, digital divide, privacy, responsibility for web content. ”
124.	1.6.1			Legal issues	“Legal issues introduced by the increasing use of computer science and its related technologies – e.g., digital content ownership, hacking, piracy.”
125.	1.6.1			Cultural issues	“Cultural issues introduced by the increasing use of computer science and its related technologies – e.g., censorship, network restrictions, cyberbullying.”
126.	1.6.1			Environmental issues	“Environmental issues introduced by the increasing use of computer science and its related technologies – e.g., fossil fuels, energy usage, hazardous materials.”
127.	1.6.1			Privacy issues	“Privacy issues introduced by the increasing use of computer science and its related technologies – e.g., always-on, voice-activated devices; CCTV; social media; GPS tracking.”
128.	1.6.1			The Data Protection Act 2018	“Legislation that protects individuals from the unreasonable use of their personal data. Updated in 2018 to cover the requirements of the General Data Protection Regulation (GDPR).”
129.	1.6.1			Computer Misuse Act 1990	“Legislation that defines electronic vandalism, unauthorised access to computer systems and theft of information.”
130.	1.6.1			Copyright Design and Patents Act 1998	“Legislation that gives creators of literary, dramatic, musical and artistic works the right to control how their material can be used.”
131.	1.6.1			Software licences	“A set of binding legal terms that often come with a commercial software application and dictate how you can use it – e.g., personal use, company use, etc.
132.	1.6.1			Open source	“Users can modify and distribute the software. Can be installed on any number of computers. Support provided by the community. Users have access to the source code. May not be fully tested.”
133.	1.6.1			Proprietary	“Users cannot modify the software. Copyright protected. Usually paid for. Licensed per user or per computer. Support provided by developers. Users do not have access to the source code. Fully tested and supported by developers.”

134.	2.1.1	2.1 Algorithms	Computational thinking	Computational thinking	“The thought processes behind formulating a problem and expressing its solution(s) so that a human or machine can effectively carry it out.”
135.	2.1.1			Abstraction	“The process of separating ideas from specific instances of those ideas at work. Computational structures are defined by their meanings while hiding away the details of how they work. Abstraction tries to factor out details from a common pattern so programmers can work close to the level of human thought, leaving out details that matter in practice but are immaterial to the problem being solved.”
136.	2.1.1			Decomposition	“The process by which a complex problem or system is broken down into parts that are easier to conceive, understand, program and maintain.”
137.	2.1.1			Algorithmic thinking	“A way of getting to a solution by identifying the steps required.”
138.	2.1.2		Designing, creating and refining algorithms	Problem inputs	“Any information or data that is fed into a system.”
139.	2.1.2			Problem processes	“Anything that happens to data while a system is running – e.g., calculations.”
140.	2.1.2			Problem outputs	“Any information or data that leaves a system.”
141.	2.1.2			Structure diagram	“A diagram that looks like an upside-down tree with one node at the top (root) and many below. Used when designing solutions to problems to help break a large problem down into a number of smaller parts.”
142.	2.1.2			Pseudocode	“A language-independent description of the steps of an algorithm. Intended for humans to express and design algorithms before coding.”
143.	2.1.2			Flowchart	“A method of designing algorithms using symbols before coding.”
144.	2.1.2			Trace table	“A technique used to test algorithms and ensure that no logical errors occur while the algorithm is being processed. The table usually has a column for each variable. Each row shows how the various values change as the algorithm runs.”
145.	2.1.3		Searching and sorting algorithms	Searching algorithms	“An algorithm that attempts to find a specific value in a data set.”
146.	2.1.3			Binary search	“Efficient search method that only works if a file’s records are arranged in sequence. Involves accessing the middle record in the file, determining whether the target record has been found and, if not, whether the target record is before or after the mid-point. The process is repeated on the part of the file where the target record is expected to be until it is found.”
147.	2.1.3			Linear search	“Examining each entry in a file in turn until the target record is found or the end of the file is reached. Unless the file is arranged in a useful order, a serial search must be used.”
148.	2.1.3			Sorting algorithm	“An algorithm that attempts to sort an unordered set of values.”
149.	2.1.3			Bubble sort	“Simple and popular with inexperienced programmers but inefficient for sorting large amounts of data, as the length of time it takes to execute correlates to the square of the number of items – e.g., if a list of 10 items takes 1ms to sort, 100 items will take 100ms.”
150.	2.1.3			Merge sort	“Divide-and-conquer algorithm created by John von Neumann. First, the list is divided into the smallest unit, known as an element. Each element is compared with the adjacent list with a view to sorting the records and merging the two lists back together.”
151.	2.1.3			Insertion sort	“A simple sorting algorithm that builds the final sorted array/list one item at a time. Less efficient with large lists than advanced algorithms like quicksort, heapsort or merge sort.”

152.	2.2.1	2.2 Programming fundamentals	Programming fundamentals	Variable	“A value that can change depending on conditions or information passed to the program.”
153.	2.2.1			Constant	“A value that cannot be altered by the program during normal execution.”
154.	2.2.1			Operator	“Tells a program how to manipulate or interpret values. Categories of operators you need to know about are arithmetic, Boolean and comparison.”
155.	2.2.1			Assignment	“Giving a variable or constant a value (e.g., counter = 0).”
156.	2.2.1			Programming construct	“Lines/blocks of code that perform a certain function. The three basic programming constructs are sequence, selection and iteration.”
157.	2.2.1			Sequence	“One of the three basic programming constructs. Instructions that are carried one after the other in order.”
158.	2.2.1			Selection	“One of the three basic programming constructs. Instructions that can evaluate a Boolean expression and branch off to one or more alternative paths.”
159.	2.2.1			Count-controlled iteration	“An iteration that loops a fixed number of times. A count is kept in a variable called an index or counter. When the index reaches a certain value (the loop bound) the loop will end. Count-controlled repetition is often called definite repetition because the number of repetitions is known before the loop begins executing.”
160.	2.2.1			Condition-controlled iteration	“A way for computer programs to repeat one or more steps depending on conditions set either a) initially by the programmer or b) by the program during execution.”
161.	2.2.1			Arithmetic operator	+ - / * ^ “Used in mathematical expressions (e.g., num1 + num2 = sum).”
162.	2.2.1			Boolean operator: AND	“A logical operator used within a program. Only returns TRUE if both values being compared are TRUE.”
163.	2.2.1			Boolean operator: OR	“A logical operator used within a program. Returns TRUE as long as either value being compared is TRUE.”
164.	2.2.1			Boolean operator: NOT	“A logical operator used within a program. Returns FALSE if the input is TRUE and returns TRUE if the input is FALSE.”
165.	2.2.1			Comparison operator: ==	“Equal to.”
166.	2.2.1			Comparison operator: !=	“Not equal to.”
167.	2.2.1			Comparison operator: <	“Less than.”
168.	2.2.1			Comparison operator: <=	“Less than or equal to.”
169.	2.2.1			Comparison operator: >	“Greater than.”
170.	2.2.1			Comparison operator: >=	“Greater than or equal to.”
171.	2.2.1			Arithmetic operator: +	“Addition.”
172.	2.2.1			Arithmetic operator: -	“Subtraction.”
173.	2.2.1			Arithmetic operator: *	“Multiplication.”
174.	2.2.1			Arithmetic operator: /	“Real division.”
175.	2.2.1			Arithmetic operator: MOD	“Integer division. MOD outputs the remainder left over after division – e.g., 10 MOD 3 = 1.”
176.	2.2.1			Arithmetic operator: DIV	“Integer division: DIV outputs the number of times a number fits into another number – e.g., 10 DIV 3 = 3.”
177.	2.2.1			Arithmetic operator: ^	“Exponent.”

178.	2.2.2	2.2 Programming fundamentals	Data types	Data type	"The basic data types provided as building blocks by a programming language. Most languages allow for more complicated, composite types to be constructed from basic types recursively – e.g., char, integer, float, Boolean. As an extension, a string data type is constructed behind the scenes of many char data types."
179.	2.2.2			Integer	"A data type used to store positive and negative whole numbers."
180.	2.2.2			Real	"A data type used to store an approximation of a real number in a way that can support a trade-off between range and precision. Typically, a number is represented approximately to a fixed number of significant digits and scaled using an exponent."
181.	2.2.2			Boolean	"Used to store logical conditions – e.g., TRUE/FALSE, ON/OFF, YES/NO, etc."
182.	2.2.2			Character	"A single alphanumeric symbol."
183.	2.2.2			String	"A sequence of alphanumeric characters and/or symbols – e.g., a word or sentence."
184.	2.2.2			Casting	"Converting a variable from one data type to another. For example, a variable entered as a string needs to be an integer for calculation – age = INPUT("Enter your age: ") age = INT(age)."
185.	2.2.3		Additional programming techniques	String manipulation	"Commands and techniques that allow you to alter and extract information from textual strings – e.g., .length .substring(x, i) .left(i) .right(i) .upper .lower ASC(…) CHR(…)."
186.	2.2.3			File handling: Open	"File handling is the process of dealing with input to and from files. Files first have to be opened, creating a handle to the file and allowing reading and writing."
187.	2.2.3			File handling: Read	"Once a file has been opened, it is possible to use commands to read its contents and return them to a program."
188.	2.2.3			File handling: Write	"Once a file has be opened it is possible to use commands to write data to the file from a program."
189.	2.2.3			File handling: Close	"When a file is no longer in use, closing it releases the file handle and breaks the connection between the file and a program."
190.	2.2.3			Record	"A data structure consisting of a collection of elements, typically in fixed number and sequence and indexed by name. Elements of records may be called fields. The record is a data type that describes such values and variables. Most modern languages allow programmers to define new record types, as well as specifying the data type of each field and an identifier by which it can be accessed."
191.	2.2.3			SQL	"The language and syntax used to write and run database queries."
192.	2.2.3			SQL command: SELECT	"A SQL keyword used query (retrieve) data." SELECT Name, Age, Class FROM Students_table WHERE Gender = "Male"
193.	2.2.3			SQL command: FROM	"A SQL keyword used to signify which table(s) are included in a query." SELECT Name, Age, Class FROM Students_table WHERE Gender = "Male"
194.	2.2.3			SQL command: WHERE	"A SQL keyword used to filter query results." SELECT Name, Age, Class FROM Students_table WHERE Gender = "Male"
195.	2.2.3			Array	"A set of data items of the same type grouped together using a single identifier. Each item is addressed by its variable name and a subscript."
196.	2.2.3			Sub-programs	"A block of code given a unique identifiable name within a program. Supports code reuse and good programming technique."
197.	2.2.3			Procedure	"A block of code within a program that is given a unique, identifiable name. Can take upwards of zero parameters when it is called. Should be designed and written to perform a task or action that is clearly indicated by its name."
198.	2.2.3			Function	"A block of code within a program that is given a unique identifiable name. Can take upwards of zero parameters when it is called and should return a value. Should be designed and written to perform a task or action that is clearly indicated by its name."
199.	2.2.3			Random number generation	"Most programming languages have built-in functions or libraries that allow you to easily generate random numbers. Creating truly random numbers is actually rather difficult for a computer, and these algorithms are quite complex."

200.	2.3.1	2.3 Producing robust programs	Defensive design	Defensive design	"The practice of planning for contingencies in the design stage of a project."
201.	2.3.1			Anticipating misuse	"Considering how an end user might accidentally or deliberately break a program and writing additional code to handle these situations."
202.	2.3.1			Authentication	"Verifying a user's identity before they can use a system. Strong passwords over a certain length with symbols and mixed-case letters are advised."
203.	2.3.1			Input validation	"Ensuring data input by a user meets specific criteria before processing. Range check (e.g., 1 – 31); type check (e.g., a number, not a symbol); presence check (e.g., data has been input); format check (e.g., a postcode is written LLN(N) NLL). "
204.	2.3.1			Maintainability	"Techniques and methods that make code easier to debug, update and maintain."
205.	2.3.1			Naming conventions	"Many programmers use defined naming conventions for variables, contents and procedures. Camel case is a popular one used in the industry where the first word of an identifier uses all lower case and all subsequent words start with a capital letter – e.g., studentsFirstName."
206.	2.3.1			Indentation	"Makes it easier to see where structures begin and end. Conditions, iterations and code inside procedures and functions should be indented."
207.	2.3.1			Commenting	"Used to explains sections of code. Ignored by the compiler."
208.	2.3.2		Testing	Testing	"Assessing the performance and functionality of a program under various conditions to make sure it works. Programmers need to consider all the devices the program could be used on and what might cause it to crash."
209.	2.3.2			Iterative testing	"Each module of a program is tested as it is developed."
210.	2.3.2			Final/terminal testing	"Checking that all the modules of a program work together as expected and the program meets the expectations of users with real data."
211.	2.3.2			Syntax error	"Rules of the language have been broken, so the program will not run. Variables not being declared before use. Incompatible variable types (e.g., sum = A); using assignments incorrectly (e.g., 2 + 2 = x); keywords misspelt (e.g., PRNT("Hello"))."
212.	2.3.2			Logical error	"The program runs but does not give the expected output. Division by zero. Infinite loop. Memory full. File not found."
213.	2.3.2			Test data	"Values used to test a program – normal, boundary and erroneous."
214.	2.3.2			Test data: Normal	"Data supplied to a program that is expected. Using a program written to average student test scores as an example, if allowed scores are 0 – 100, normal test data would include all the numbers within that range."
215.	2.3.2			Test data: Boundary	"Data supplied to a program designed to test the boundaries of a problem. Using a program written to average student test scores as an example, if allowed scores are 0 – 100, boundary test data could be -1, 0, 1, 99, 100 and 101."
216.	2.3.2			Test data: Invalid	"Data of the correct type but outside accepted validation limits. Using a program written to average student test scores as an example, if allowed scores are 0 – 100, invalid test data could be -5, 150, etc."
217.	2.3.2			Test data: Erroneous	"Data of the incorrect type that should be rejected. Using a program written to average student test scores as an example, if allowed scores are 0 – 100, erroneous data might be the string "hello", the real number 3.725, etc."
218.	2.4.1	2.4 Boolean logic	Boolean logic	Logic diagram	"A method of expression Boolean logic in a diagram using a set of standard symbols that represent the various logic gates – AND, NOT, OR, NAND, etc."
219.	2.4.1			Logic gate	"A symbol in a logic diagram that represents a single gate – e.g., AND, OR, NOT."
220.	2.4.1			Logic gate: AND	"Accepts two inputs and produces one output. Both inputs must be TRUE (1) for the output to be TRUE (1) – otherwise, the output will be FALSE (0)."
221.	2.4.1			Logic gate: OR	"Accepts two inputs and produces one output. At least one input must be TRUE (1) for the output to be TRUE (1) – otherwise, the output will be FALSE (0)."
222.	2.4.1			Logic gate: NOT	"Accepts one input and produces one output. If the input is TRUE (1), the output will be FALSE (0). If the input is FALSE (0), the output will be TRUE (1)."
223.	2.4.1			Truth table	"A notation used in Boolean algebra to define the output of a logic gate or logic circuit for all possible combinations of inputs."
224.	2.5.1	2.5 Programming languages and IDEs	Languages	High-level language	"Designed to allow the expression of a computer program in a way that reflects the problem being solved rather than the details of how the solution is produced. One-to-many."
225.	2.5.1			Low-level language	"Close to machine code and closely related to the design of the machine. One-to-one."
226.	2.5.1			Translator	"Takes a program written in one programming language and converts it to another."
227.	2.5.1			Compiler	"Translates high-level language source code into a computer's machine code."
228.	2.5.1			Interpreter	"Translates and executes a program one statement at a time."
229.	2.5.2		The Integrated Development Environment	IDE	**Integrated Develop Environment:** "A software application that provides comprehensive facilities for software development. Normally consists of a source code editor, build automation tools and a debugger."
230.	2.5.2			IDE: Error diagnostics	"IDE tools that provide detailed feedback on errors in code."
231.	2.5.2			IDE: Run-time environment	"A configuration of hardware and software. Includes the CPU type, operating system and any runtime engines or system software required by a particular category of application."

Notes page

Notes page

Y11 GCSE Exam Dates

Y11 Mock(s): _____

Y11 PPE(s): _____

Final GCSE(s): _____

_____

_____

Success Programme Sessions:

_____

_____

_____

Revision Guide (if applicable):

_____

_____

_____

_____

Notes

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____