

Cambridge AS Level

Computer Science

Edited (mainly) by Li Junxi

MS
2026

Preface

This document is an aggregation of all course book knowledges and mark schemes of the Cambridge International AS & A Level Computer Science exams from 2015 to 2024 (*Update: now 2025, thanks for the contributions made by Jiazhou Wu*), edited by Li Junxi, a student from the A-Level program of Guangdong Country Garden School who got 96/100 in the actual AS Computer Science exam.

The structure of this document strictly follows the structure of Cambridge International AS & A Level Computer Science Syllabus (2024-2025), ensuring coverage of all knowledge you need to know in order to master the exam.

The **blue** texts are the notes and guidances in the syllabus, which provide insights about the details of the requirements. The underlined content are ones that you must write to gain that point. *Texts like this* are my comments. **Red** contents are those no longer in the latest version of the syllabus. You can ignore them if you want. **Green** texts are used when different years have different answers towards a same question. **Orange** contents are analysed from textbook or other sources. The **[]** bracket indicates the marks of the question, namely the number of points you need to gain full mark of that question.

I sincerely hope that this document can fulfil its job helping you achieving better results in the exam. You can share it to anyone without any concerns. Just don't pretend you wrote it, thanks.

1 Information representation	5
1.1 Data Representation	5
1.2 Multimedia - Graphics, Sound	9
1.3 Compression	16
2 Communication	22
2.1 Networks Including the Internet	22
3 Hardware	47
3.1 Computers and their Components	47
3.2 Logic Gates and Logic Circuits	61
4 Processor Fundamentals	64
4.1 Central Processing Unit (CPU) Architecture	64
4.2 Assembly Language	74
4.3 Bit Manipulation	81
5 System Software	84
5.1 Operating System	84
5.2 Language Translators	93
6 Security, privacy and data integrity	99
6.1 Data Security	99
6.2 Data Integrity	106
7 Ethics and Ownership	110
7.1 Ethics and Ownership	110
8 Databases	122
8.1 Database Concepts	122
8.2 Database Management System (DBMS)	128

8.3 Data Definition Language (DDL) and Data Manipulation Language (DML)	132
9 Algorithm Design and Problem-Solving	137
9.1 Computational Thinking Skills	137
9.2 Algorithms	138
10 Data Types and structures	142
10.1 Data Types and Records	142
10.2 Arrays	143
10.3 Files	145
10.4 Introduction to Abstract Data Types (ADT)	146
11 Programming	150
11.1 Programming Basics	150
11.2 Constructs	152
11.3 Structured Programming	154
12 Software Development	158
12.1 Program Development Lifecycle	158
12.2 Program Design	161
12.3 Program Testing and Maintenance	163
13 Appendix	169
13.1 Generic Marking Principle	169
13.2 Insert for paper 2	170
13.3 Pseudocode symbols and keywords	173
13.4 Further resources	174
13.5 Epilogue	174

1 Information representation

1.1 Data Representation

1.1.1 Show understanding of binary magnitudes and the difference between binary prefixes and decimal prefixes

Understand the difference between and use:

- kibi and kilo
- mebi and mega
- gibi and giga
- tebi and tera

Binary prefix	Number	Decimal prefix	Number
kibi	1024	kilo	10^3
mebi	1024^2	mega	10^6
gibi	1024^3	giga	10^9
tebi	1024^4	tera	10^{12}

1.1.2 Show understanding of different number systems

- Use the binary, denary, hexadecimal number bases and Binary Coded Decimal (BCD) and one's and two's complement representation for binary numbers.
 - Convert an integer value from one number base/ representation to another
- **BCD [2]**
 - convert each digit to 4-bit binary
 - e.g. 97 to 1001 0111
 - **Two's complement [2]**
 - invert (one's complement)

- add one
- the largest two's complement binary number:
 - 0111 1111
- the smallest two's complement binary number:
 - 1000 0000
- **Convert number bases**
 - *You should be able to do this*

1.1.3 Perform binary addition and subtraction

Using positive and negative binary integers Show understanding of how overflow can occur

- **Addition [2~3]**
 - Convert denary / hexadecimal to binary (*if need to*)
 - Method for addition
 - Final answer
 - use (1) if overflow occurs
 - e.g.

$$\begin{array}{r}
 0010 \ 0011 \\
 + \ 0000 \ 1111 \\
 \hline
 0011 \ 0010 \\
 1 \ 111
 \end{array}$$

- **Subtraction [3]**
 - Convert numbers to two's complement
 - Adding values
 - Final answer
- **Overflow [1]**
 - The result is a larger number than can be stored in the given number of bits

- The answer cannot be represented in the number of bits available

1.1.4 Describe practical applications where Binary Coded Decimal (BCD) and Hexadecimal are used

- **BCD**

- **Benefit [2]**

- straightforward to convert to / from BCD and denary
- ... so it's less complex to encode and decode for programmers
- easier for digital equipment use BCD to display output information
- can represent monetary values exactly

- **Application [1]**

- when denary number need to be electronically coded (*A string of digits on any electronic device displaying numeric values*)
- e.g. to operate displays on a calculator where each digit is represented
- decimal fractions can be accurately represented.

- **Justification of choosing [1]**

- ... because conversion with denary is easier
- ... because visual displays only need to show individual digits
- ... because it is difficult to represent decimal values exactly in normal binary and financial transactions use only two decimal places and must be accurate, no accumulating errors

- **Hexadecimal (from textbook)**

- Application

- represent memory dump
- error code of error occurred during the execution of a program
- when the bytes contain binary numbers in the charts that define character codes (UNICODE)

1.1.5 Show understanding of and be able to represent character data in its internal binary form, depending on the character set used

Students are expected to be familiar with ASCII (American Standard Code for Information Interchange), extended ASCII and Unicode. Students will not be expected to memorise any particular character codes.

- Character set

- Define [1]

- The symbols that the computer recognises
- A list of characters recognised by the computer hardware and software

- Process single character [2]

- Each character is represented by an unique / corresponding
- binary code / integer / value

- Process (string) [2]

- Each character has its own unique code
- Each character in the word is replaced by its code
- The codes are stored in the order in the word

- ASCII [2]

- only 128 / 256 characters can be represented
- Uses values 0 to 127(or 255 if extended form) / one byte (7 or 8 bits)
- Many characters used in other languages can't be represented
- In extended ASCII the characters from 128 to 255 may be coded differently in different systems. (*not standardised*)
- **Unicode [2]**
 - Uses 16, 24 or 32 bits / two, three or four bytes
 - Unicode is designed to be a superset of ASCII
 - Designed so that most characters (in other languages) can be represented

1.2 Multimedia - Graphics, Sound

Graphics

1.2.1 Show understanding of how data for a bitmapped image are encoded

Use and understand the terms: pixel, file header, image resolution, screen resolution, colour depth/ bit depth

- **Bitmap image**
 - **Describe (represent) [2]**
 - Made up of pixels
 - Each pixel has one colour
 - Colour of each pixel stored as a binary number
 - **Describe (encode) [3]**
 - Image is made of pixels and each pixel has one colour
 - Each colour has a unique binary code
 - Code for the colour of each pixel is stored in sequence

- **Pixel [1]**
 - Smallest picture element which can be drawn
- **File header**
 - **Purpose [1]**
 - Stores data about the file contents/image/metadata
 - **Example [2]**
 - Confirmation that the file is a BMP // confirmation of the file type
 - File size
 - Location / offset of image data within the file
 - Dimensions of the image // image resolution
 - Colour depth
 - Type of compression used (if any)
- **Image resolution [1]**
 - The number of pixels per unit measurement
 - The number of pixels in an image
 - The number of pixels wide by the number of the pixels high
 - Number of pixels per row by the number of rows
- **Screen resolution [1]**
 - The number of pixels which can be viewed horizontally and vertically on the screen // or by example - A typical screen resolution is 1680 x 1080 pixels
- **Colour depth [1]**
 - The number of bits per pixel // determines the number of colours that can be represented in the image

1.2.2 Perform calculations to estimate the file size for a bitmap image

- **Calculation [2]**

- bit-depth x number of pixels
- convert to correct unit (KB, MB, etc.)

- **Why calculate [1]**

- so it is possible to estimate how many images can be stored
- to decide if it can be sent as an email attachment

- **Why actual larger than calculated [2]**

- The file will contain metadata
- The file will have a header

1.2.3 Show understanding of the effects of changing elements of a bitmap image on the image quality and file size

Use and understand the terms: pixel, file header, image resolution, screen resolution, colour depth/ bit depth

- **Increase colour depth / bit depth**

- **File size [2]**

- increased file size
 - more bits per pixel and hence more data stored

- **Image Quality [2]**

- greater range of colours
 - makes the image closer to the original / more realistic

- **Increase image resolution**

- **File size [1]**

- increase file size since there are more pixels to store

- **Image quality [2]**

- more pixels can be stored / are available
- the image is sharper / less pixelated

1.2.4 Show understanding of how data for a vector graphic are encoded

Use the terms: drawing object, property, drawing list

- **How vector graphic represent image [3]**
 - It is a series of (geometric) shapes / lines / objects
 - Stored coordinates
 - Contains a drawing list
 - Commands / formulae for creating each object individual object
 - and the attributes/property for that object
 - e.g. colour, thickness of line etc.
- **Drawing list**
 - **Define [1]**
 - All the drawing objects in an image // a list that stores the commands required to draw each object
 - **Describe [2]**
 - List of objects in the drawing
 - A list that stores the command/description/equation required to draw each object
 - Properties of each object e.g. the fill colour, line weight/colour
- **Drawing object [1]**
 - A component created using a formula
- **Property [1]**

- Data about the shapes // defines one aspect of the appearance of the drawing object

1.2.5 Justify the use of a bitmap image or a vector graphic for a given task

- **Difference between bitmap and vector graphic [4]**

- Bitmap made up of pixels
- Vector graphic store a set of instructions about how to draw the shape
- When bitmap is enlarged the pixels get bigger and it pixelates
- When vector is enlarged it is recalculated and does not pixelate
- Bitmap files are usually bigger in size because of the need to store data about each pixel
- Vector graphics have smaller file size because they contain just the instructions to create the shapes
- Bitmap images can be compressed with significant reduction in file size
- Vector graphics do not compress well because of little redundant data

- **Advantage of bitmap files**

- can be compressed (with significant reduction in file size)
- suitable for photographs /scanned images
- does not need to be 'rasterised' to display

- **Advantage of vector graphic [4]**

- Can resize without pixilation
- ... Image is drawn/recalculated with each adjustment
- Smaller file size
- ...Storing points/equations/commands etc. not individual pixels

Sound

1.2.6 Show understanding of how sound is represented and encoded

Use the terms: sampling, sampling rate, sampling resolution, analogue and digital data

- **How sound is represented by computer [3]**
 - The amplitude is recorded a set number of times a second
 - Each (instance of an) amplitude is given a corresponding binary number
 - The binary number (of each amplitude) is saved in sequence
- **Sampling**
 - **Define [1]**
 - amplitude of sound wave taken at different points in time
 - measurement of value of analogue signal at regular time intervals/ a point in time
 - **Process [3]**
 - The height of sound wave is determined
 - At set (time) intervals // by example of sensible time period
 - To get an approximation of the sound wave
 - And encoded as a sequence of binary numbers // and converted to a digital signal

- Increasing the sampling rate will improve the accuracy of the recording
- **Sampling rate [1]**
 - The number of samples taken per unit time / per second
- **Sampling resolution [2]**
 - resolution is the number of distinct values available to encode/represent each sample
 - specified by the number of bits used to store/record each sample
 - sometimes referred to as bit depth
 - usually 8 bit, 16bit, 24bit or 32 bit
- **Analogue data [1]**
 - data values that are continuously changing // variable // any value
 - the type of sound wave before it is recorded by a computer

1.2.7 Show understanding of the impact of changing the sampling rate and resolution

Including the impact on file size and accuracy

- **Higher sampling resolution**
 - **File size [2]**
 - More bits per sample
 - So more bits stored altogether, larger file size
 - **Accuracy [2]**
 - Increases the number of bits per sample // wider range of amplitudes can be stored
 - Each binary amplitude is closer to the analogue amplitude

- Digital waveform closer to original waveform
- smaller quantisation errors (*you can almost always gain point by writing this one*)
- **Benefit [1]**
 - larger dynamic ranges
 - ... as dynamic range is approximately six times the bit depth
 - more accurate representation/crisper sound quality
- **Drawback [1]**
 - bigger files
 - longer to transmit data
 - greater processing power needed
- **Higher sampling rate**
 - **Accuracy [2]**
 - Smaller time gaps between the samples
 - Makes the digital sound wave more accurate
 - Smaller quantisation errors
 - **File size [2]**
 - More samples/data are taken/recorded
 - ... so more bits are stored altogether

1.3 Compression

1.3.1 Show understanding of the need for and examples of the use of compression

- **Need to compress [2]**
 - Reduce the file size
 - Faster to transmit/download

- Original file is too large for email storage/attachment (*if email*)
- **Need to compress image on server [3]**
 - Download image in less time
 - ... will take less of bandwidth
 - Image will take up less space on storage device (*user's or server's, depending on which one question asks about*)
 - ... therefore can store more images
 - ... and more space for other files

1.3.2 Show understanding of lossy and lossless compression and justify the use of a method in a given situation

- **Lossless compression**
 - **Describe**
 - lossless designed to lose none of the original detail / allows original file to be recreated exactly
 - lossless technique based on some form of replacement
 - example of replacement: RLE, FLAC.
 - maximum compression about 50%
 - **Justify choosing (video streaming) [3]**
 - Original recording may not have been made in high resolution
 - Could be streaming to high bandwidth devices
 - The reduction in size is sufficient for the receiving device
 - Viewers do not want any loss of quality
- **Lossy compression**

- *In general, when justifying using lossy, you can always write these two points:*
 - The change may not be noticeable
 - produce larger decrease in file size compared to lossless
- **Justify choosing (audio) [3]**
 - Lossy create smaller file than lossless
 - The recording may need significant reduction in size
 - Lossy removes detail which can be lost without people noticing
 - By example e.g. reduction in sound quality will not be noticed
- **Justify choosing (for transmission) [3]**
 - File takes less storage space on receiving device than if lossless compression was used
 - The file is faster to transmit
 - Use less bandwidth
 - ... so buffering is reduced
- **Justify choosing (video streaming) [3]**
 - Loss of quality will not be noticed
 - Needs to be viewed in real time so less bandwidth needed if file size smaller
 - Smaller file size will reduce buffering so the video will play more smoothly
 - Viewers may watch on different devices, so may not need high quality resolution

1.3.3 Show understanding of how a text file, bitmap image, vector graphic and sound file can be compressed

Including the use of run-length encoding (RLE)

- **Run-length encoding [3]**

- Lossless method of compression
- Reduced (the physical size) of a string of adjacent, identical characters / pixels / bytes, etc.
- The repeating string (a run) is encoded into two values
- one value represents the number of (identical) characters in the run (the run count)
- The other value is the code of the character / colour code of pixel etc. in the run (the run value)
- The run value and run count combination may be preceded by a control character.
- Any valid example given

- **Why RLE may not reduce file size [3]**

- RLE stores a value and the number of times it occurs consecutively
- A file may not have many sequences of same value
- It would need to store each value and then count/number 1 which adds data
- e.g. Red-Green-Blue would become Red 1 Green 1 Blue 1

- **Compression of sound**

- **Lossy (describe in general) [1~2]**

- may result in loss of detail compared to original file/ does not allow original file to be re-created exactly
- make decision about what parts of sound are important and discards other information.
- ...then applies lossless technique for further reduction
- lossy can reduce about 10%
- example: jpeg, mp3

- **Lossy (describe method) [2]**

- Decrease sample rate
- ... fewer samples/readings/measurements stored per second // fewer bits per second stored

- Decrease sample resolution
- ... fewer bits per sample/reading/measurement // each sample has fewer bits

- Sound outside of human hearing range is removed
- ... fewer measurements are stored / decreases the number of possible binary values so fewer bits are stored

- **Lossless [2]**
 - Reduce amplitude to only the range used
 - ... limited amplitudes mean fewer bits per sample
 - Run-length-encoding
 - ... Where consecutive sounds are the same, record the binary value of the sound and the number of times it repeats
 - Record the changes instead of the actual sounds
 - *Huffman coding (appeared 2020)*

- **Compression of text**
 - **Why lossy can't be used [2]**
 - None of the original data can be lost / deleted
 - The file would be corrupted // the file can't be opened
 - **Lossless [2]**
 - Can use run-length encoding...
 - Identifies groups of repeated characters ...

- ... replaces them with one copy of the character and the number of times it occurs
- **Compression of bitmap Image**
 - **Lossy [4]**
 - Reduce bit depth
 - ... reduces the number of bits per colour / pixel which means each pixel has fewer bits
 - Reduce colour palette // reduce number of colour
 - ... fewer colours means fewer bits needed to store each colour
 - Reduce image resolution
 - ... fewer pixels per unit measurement means less binary to store
 - **Justify lossy [2]**
 - reduce more file size than lossless
 - so significantly less bandwidth
 - so buffering is reduced
 - **Lossless [3]**
 - Run-length encoding
 - Identifies consecutive repeating pixels of the same colour
 - Stores the colour /pattern and the number of times it repeats

2 Communication

2.1 Networks Including the Internet

2.1.1 Show understanding of the purpose and benefits of networking devices

2.1.2 Show understanding of the characteristics of a LAN (local area network) and a WAN (wide area network)

- **LAN**

- **Characteristics [2]**

- Small geographical area
 - The infrastructure is privately owned // not controlled by external organisations // No leasing external infrastructure / transmission media // does not use internet to transmit within the building etc.

- **WAN**

- **Characteristics [2]**

- Cover large geographical area
 - External / public infrastructure is used
 - Non-dedicated hardware

2.1.3 Explain the client-server and peer-to-peer models of networked computers

- Roles of the different computers within the network and subnetwork models
- Benefits and drawbacks of each model
- Justify the use of a model for a given situation
- **Client-server model**
 - **Describe [2]**
 - at least one computer used to 'serve' ...
 - ... other computers are referred to as 'clients'
 - server provides services / applications etc. ...
 - ... which may be requested by clients
 - **Describe with example of downloading a file [2]**
 - The file is made available from a web/email/FTP server
 - The user's browser is the client software
 - The client requests the file from the server
 - The desired file is returned to the client computer
 - **Example application [2]**
 - Sending and receiving email
 - Print server
 - File server
 - etc.
 - **Benefit of client-server model [2]**
 - files and resources are centralised
 - creation of security / manage security
 - user needs user name and password to access the network

- centralised back-up
- intranet capability
- Internet monitoring
- clients can be less powerful machines, therefore is less expensive to buy
- saving resources on server reduces the burden on the client
- **Describe client-side scripting [2]** *(no longer in syllabus)*
 - The user's web browser is the client software
 - The requested web page has program code / script embedded within it
 - The code is interpreted by the web browser
- **Describe server-side scripting [4]** *(no longer in syllabus)*
 - The browser requests the web page
 - The web server accesses the page
 - The web server processes / executes the code
 - The web server produces the HTML for the page / generates the web page
 - The web server returns the web page to the client
 - The client browser displays this web page
- **Peer-to-peer model**
 - **Describe [2]**
 - All computer are of equal status
 - Each computer provides access to resources and data // data is distributed
 - Computer can communicate and share resources
 - Each computer is responsible for its own security
 - **Advantage (analysed from textbook)**

- Avoid congestion when many clients are simultaneously attempting to download files
- Parts of a file can be downloaded separately
- Parts are available from more than one host.
- **Drawback [4]**
 - Reduced security // no central management of security
 - only as secure as the weakest computer on the network
 - each computer is at risk from viruses from other computers
 - No central management of backup
 - if the data from one computer is not backed up it is lost to all of them
 - No central management of files/software
 - consistency may be difficult to maintain
 - each computer may have different software from the users
 - Individual computers may respond slower
 - because they are being accessed by other computers
 - In order to share files etc. all the computers involved need to be switched on
 - otherwise the files etc. may not always be available

2.1.4 Show understanding of thin-client and thick-client and the differences between them

- **Thin-client**
 - **Describe [2]**

- Server performs all processes required by the task and/or data storage
- Clients only send requests to the server and displays the returned results
- **Characteristics [2]**
 - Data is not stored on client computer
 - Client computer is reliant on access to server
 - Client computer heavily reliant on network/internet connection
 - Client computer requires few local resources/memory
 - Client computer performs minimal functions/processes
- **Thick-client**
 - **Describe [2]**
 - The server performs minimal / some processing for the client
 - The client also do most of their own processing/work independently // most of the resources are installed locally

2.1.5 Show understanding of the bus, star, mesh and hybrid topologies

- Understand how packets are transmitted between two hosts for a given topology
- Justify the use of a topology for a given situation
- **Bus topology (analysed from textbook)**
 - **Identify**
 - only one link, shared by a number of end-systems // multi-point connection

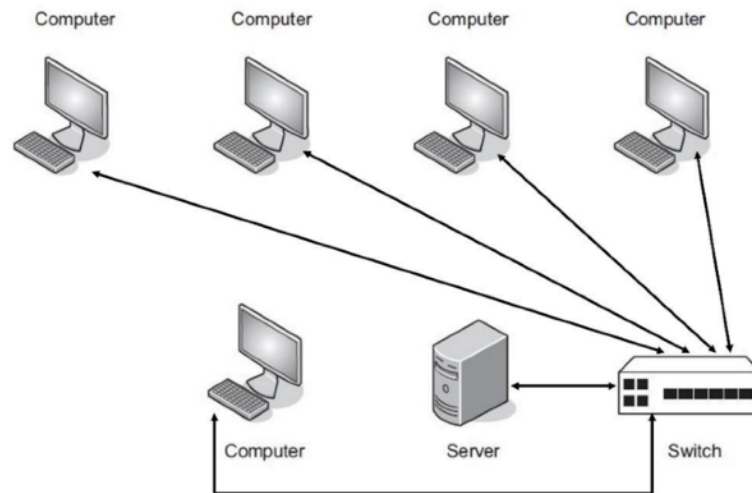
- no direct connection between any pair of end-systems
- **How data is transmitted**
 - message must be broadcast to all end-systems
 - even though it might only be intended for one
- **Star topology**
 - **Identify [1]**
 - Devices are connected directly to the router independently // all devices are only connected to the router
 - **How data is transmitted [2]**
 - The data from the sending laptop is transmitted to the router
 - The data has address of recipient
 - The router determines the recipient's destination address
 - ... using a routing table
 - The router transmits data directly / only to **intended** recipient
 - **Advantage (against bus) [2]**
 - More resilient to faults
 - ... because there is no single cable and leads to less disruption to network
 - Higher performance as fewer collisions
 - ... because each device is only connected to the central device
 - Easier to add nodes
 - ... because each device connects directly to the central device

- Easier to find fault

- Diagram [2]

1 mark per bullet point, max 2 marks

- All computers directly connected to switch and no other connections
- Switch connected directly to server



- Mesh topology

- Describe [2]

- All computers are connected to at least one other device
- There are multiple routes between devices
- The computers can act as relays, passing packets on towards the final destination

- Advantage (against bus) [2]

- If one line goes down there are multiple routes available
- Improved security as not using one main line
- No/fewer collisions
- New nodes can be added without interruption or interfering with other nodes
- More secure because data is sent over a dedicated connection

- Hybrid topology *(analysed from textbook)*

- Different LANs connected, each with different topologies
- Special connecting device is needed
- able to connect a new topology LAN to existing LANs where it is not sensible or not possible to use the existing topology for the new LAN.

2.1.6 Show understanding of cloud computing

- Including the use of public and private clouds
- Benefits and drawbacks of cloud computing
- **Cloud computing**
 - **Define [1]**
 - Accessing a service / files / software on a remote server
 - **Benefit [2]**
 - Can be accessed anywhere with Internet access
 - Do not need to install security // security might be better
 - Do not need to perform backups
 - Do not need to buy specific software/hardware
 - Can easily share documents
 - Can have multiple people working on the same document
 - **Drawbacks [1]**
 - You cannot access it if no internet access
 - Reliant on someone else to backup
 - Reliant on someone else for security // can have poorer security
 - Cannot access if server goes down
 - **Public cloud**

- **Describe [1]**
 - Computing services offered by 3rd party provider over the public internet
 - Public is open/available to anyone with the appropriate equipment/software/credentials
- **Private cloud**
 - **Describe [1]**
 - Computing services offered either over the internet or a private internal network
 - Only available to selected users not the general public
 - Private is a dedicated/bespoke system only accessible for/from the organisation
 - **Benefit [3]**
 - Not reliant on a third party
 - ... gives greater control over security/privacy
 - ... gives greater control over backup
 - Storage can be tailored/scalable to company requirements // an example e.g. the amount of storage accessible / facilitating the sharing of files
- **Storing data on cloud**
 - **Advantage [2]**
 - Can be free (for small quantities)
 - No need for separate (high capacity) storage devices // saving storage on existing devices
 - Can access data from any computer with internet access.
 - Most cloud data services will have built-in backup/disaster recovery

- Security could be better
- Can easily increase capacity
- Data can be easily shared
- **Disadvantage [2]**
 - Cannot access files without internet connection
 - No control over security (*... since data is stored on a remote infrastructure*)
 - No control over backups (*prioritise these three points*)
 - It can be more expensive in the long term // recurring costs
 - ... as cloud provider charges must be paid
 - There could be a limit to the amount of storage unless paid for
 - There could be compatibility/access issues
 - There could be issues with the company offering cloud services

2.1.7 Show understanding of the differences between and implications of the use of wireless and wired networks

Describe the characteristics of copper cable, fibre-optic cable, radio waves (including WiFi), microwaves, satellites

- **Copper cable [1]**
 - Carries data as electrical signals and can consist of a twisted pair
- **Fibre-optic [1]**
 - Transmit data as pulses of light
- **Radio waves / microwaves [1]**
 - Transmits electromagnetic waves on different frequencies

- **Comparing copper cable and fibre-optic cable [4]**

- Fibre-optic cables have greater bandwidth
- Fibre-optic cables need less signal boosting // can travel over longer distance than copper cable before repeaters are needed
- Fibre-optic cables have greater security (more difficult to 'tap' into)
- Fibre-optic cables have smaller risks of interference
- It is easier to make terminations using copper cabling
- Fibre optic is more prone to damage than copper cable (*may break if bended through large angle*)
- fibre-optic cabling is lighter in weight (easier to install)
- fibre-optic cables consume less power
- copper cabling is less expensive to install
- copper cable is easier to install because it is more flexible
- the expertise in use of copper cabling is more extensive
- has been around for years ... so very little "unknown" about installations using this type of cabling.

- **Wireless network**

- **Benefits [3]**

- Devices can be more mobile as they do not have to be connected to cable
- Easier to set up // no cables need to be installed
- Add additional devices is easier
- Many different types of device can be connected at the same time

- **Drawback [1]**

- Easier to hack
- interference

- Signal degrades quickly
- **Wired network**
 - **Benefit [1]**
 - Faster connection / communication / transmission of data
 - More consistent transmission speed
 - Improved security
 - **Drawback [1]**
 - Expensive to set-up / maintain
 - Disruption to the dedicated line would leave no alternative
- **Satellite [1]**
 - A communication device in Earth's orbit that receives and transmits data

2.1.8 Describe the hardware that is used to support a LAN

Including switch, server, Network Interface Card (NIC), Wireless Network Interface Card (WNIC), Wireless Access Points (WAP), cables, bridge, repeater

- **Switch [1]**
 - To allow two or more devices to communicate with one another
 - To connect individual devices to each other
 - To receive transmissions and forward them to their destination / intended recipient
 - store MAC addresses
- **Server [1]**
 - A device or software that provides a specific function for computers using a network.

- **Network Interface Card(NIC) [1]**
 - Provide physical connection between computer and network
 - Has unique MAC address to identify a device
- **Wireless Network Interface Card (WNIC)**
 - **Describe [1]**
 - Hardware component that allows a device to connect to a wireless network // Provides a MAC address to the device to identify it on the wireless network
 - **Function [4]**
 - Provides interface to wireless network
 - ... as an antenna
 - Receives analogue radio waves
 - ... convert them to digital/binary
 - Takes digital/binary input and converts to analogue waves
 - ... sends the radio waves via the antenna
 - Checks incoming transmissions for correct MAC / IP address
 - ... ignore transmissions not intended for it
 - Encrypts / encodes the data
 - Decrypts / decodes the data
- **Wireless Access Points (WAP)**
 - **Describe [1]**
 - Hardware component that provides radio communication from the central device to nodes on the network
 - **Purpose [1]**
 - To allow connection of devices (to the central device) using radio signals / Wi-Fi

- To allow the central device to send / receive radio signals / Wi-Fi signals
- To allow wireless enabled devices to connect to a wired network
- **Cables** (*analysed from textbook*)
 - act as transmission medium, carry data between different devices
- **Bridge [1]**
 - To connect two LANs / segments with the same protocol
 - To transmit data between two networks with the same protocol
- **Repeater [1]**
 - Restores the digital signal so it can be transmitted over greater distances

2.1.9 Describe the role and function of a router in a network

- **Router**
 - **Role [3]**
 - Connects two (or more) networks
 - Can connect a network to a WAN // act as the single access point for ...
 - Receives packets and forwards towards the destination
 - ... using IP of the destination
 - Assigns private IP address
 - Operates between similar networks // networks using the same protocol
 - Can be used to segment a network

- **Functions [3]**

- Receive packets from devices or the internet
- To forward / route packets to the destination
- Find the destination of the packet
- Assign / allocate private IP addresses to devices on LAN
- Store / update / maintain a routing table
- Find the most efficient path to the destination
- Maintain a table of MAC and IP addresses

- **Gateway [3]** (not directly in syllabus)

- Connect two (or more) networks
- Can connect a network to a WAN // act as a single accessing point for...
- Receives packets and send packets towards the destination
- ... using the IP address of the destination
- Assigns private IP address
- Connects two dissimilar networks // networks that use different protocols

2.1.10 Show understanding of Ethernet and how collisions are detected and avoided

Including Carrier Sense Multiple Access / Collision Detection (CSMA / CD)

- **Describe ethernet [3]**

- A protocol (suite)
- For data transmission over standard / universal wired / cabled network connections
- Uses Carrier Sense Multiple Access / Collision Detection (CSMA/CD)

- Data is transmitted in frames
- ... each frame has a source and destination (IP/MAC) address
- ... and error checking dat (so damaged frames can be resent)
- **CSMA/CD**
 - **Operation [4]**
 - A workstation / node (wishing to transmit) listens to the communication channel
 - ... data is only sent when the channel is free // ... if channel is free data is sent
 - Because there is more than one computer connected to the same transmission medium
 - ... two workstations can start to transmit at the same time, causing a collision
 - If a collision happens, the workstations send a (jamming) signal / abort transmission
 - ... and each waits a random amount of time before attempting to resend
 - **Drawback [2]**
 - Random time increased each time so can be infinite waiting
 - May be constant jamming signal so nothing ever sends
 - Certain nodes cannot be prioritised
 - High power consumption
 - Only suitable for short distance network // limited distance
 - Not scalable // more nodes means exponentially longer waiting times

2.1.11 Show understanding of bit streaming

Methods of bit streaming, i.e. real-time and on-demand

Importance of bit rates broadband speed on bit streaming

- Definition [2]

- sequence of digital signals / bits
- over a communication path / internet
- transfer data at high speed
- requires fast broadband connection
- requires some form of buffering
- bits arrive in the same order as sent

- Advantage of bit streaming [2]

- no need to wait for a whole file to be downloaded
- no need to store larger files on user's computer
- allow on demand playback (if on-demand bit streaming)
- ... and no specialist software needed for playback in browser

- Disadvantage of bit streaming [2]

- video stops if slow internet / broadband speed slow
- video stops if inadequate buffering capacity
- loss of Internet means can't access films / files
- viruses can be downloaded from the websites

- On-demand bit streaming [2]

- digital video tape or digital files converted to bit streaming format for broadcasting on the net; this is known as encoding, these encoded streaming video files are then uploaded to a dedicated server
- a link for the encoded video is placed on a website
- a user clicks on the link to download the encoded streaming video; the streamed video is

- then broadcast to the user as and when they need it
- can be paused / can go back and rewatch / fast-forward, etc.
- **Real-time bit streaming [2]**
 - an event is captured live with a video camera
 - video camera connected to a computer
 - video signal is converted to streaming media files(encoded) on the computer
 - encoded feed uploaded from computer to a dedicated streaming server via cable, DSL, or a high-speed internet connection
 - server then sends the live images to all users requesting it as real-time video streaming
 - cannot be paused, etc.
- **Difference between real-time and on-demand [2]**
 - Real-time is direct from the source whereas on-demand is pre-recorded/downloaded to view later
 - Real-time cannot be rewatched, on-demand can be paused, rewatched etc.
 - Real-time plays continually, on-demand downloads sections/ blocks and cannot play until next section is downloaded
- **Describe how video is sent using bit streaming [4]**
 - Video is compressed before transmitting
 - The video is transmitted continuously as series of bits
 - The video is hosted on a media server
 - On download, the server sends the data to a buffer on the client computers // The buffer stores the data from the server
 - The recipient / user's software receives bit stream from the buffer

2.1.12 Show understanding of the differences between the World Wide Web (WWW) and the internet

- **Internet [2]**

- Global network of interconnected computer networks
- uses TCP/IP protocol
- Internet stands for Interconnected Networks
- Infrastructure provided by ISP

- **World Wide Web(WWW)**

- is a collection of web pages/documents
- ... stored on websites
- http/https protocol
- web pages written in HTML
- URLs specify the location of the web pages
- web documents accessed using browsers.

2.1.13 Describe the hardware that is used to support the internet

Including modems, PSTN (Public Switched Telephone Network), dedicated lines, cell phone network

- **Modem**

- **Purpose [1]**

- To connect device to the Internet over a telephone line
- Conversion between digital signal and analogue signal

- **PSTN**

- **Describe [2]**

- The PSTN consists of many different types of communication lines

- ... digital data may need to be converted into analogue signal
- Data is transmitted in both direction at the same time // (full) duplex data transmission
- Communication passes through different switching centers.
- **Characteristics**
 - dedicated channel used between two points for the duration of the call
 - connection maintained throughout telephone call
 - line remain active even during power outage
 - no encoding and compression
- **Cell phone network**
 - **Explain [4]**
 - land is spilt into cells for maximum line of sight
 - each cell has a tower with an antenna which receives and transmits data
 - data is transmitted between the tower and the phone
 - data transmission is wireless using low power radio signals
 - multiple devices can communicate simultaneously with the same tower
- **Dedicated lines**
 - **Describe [1]**
 - Used to provide a direct / private connection which therefore provides faster transmission

2.1.14 Explain the use of IP addresses in the transmission of data over the internet

Including:

- format of an IP address including IPv4 and IPv6
 - use of subnetting in a network
 - how an IP address is associated with a device on a network
 - difference between a public IP address and a private IP address and the implications for security
 - difference between a static IP address and a dynamic IP address
- **Full name of IP [1]**
 - Internet protocol
 - **Purpose of IP address[2]**
 - Gives each device on a network an identifier // IP address used to locate a device on a network
 - Each address is unique within the network
 - Allows a device/gateway/node to send data to the correct destination / a specific device/gateway/node
 - **IPv4 format [1]**
 - Contains 4 groups of digits. Each group is represented in 8 bits (0~255) and the groups are separated by full stops.
 - **IPv6 format [1]**
 - Contains 8 groups of digits. Each group is represented in 16 bits (0 ~ FFFF). Multiple groups that only contain zeros can be replaced with a :: // double colon.
 - **Subnetting**
 - **Purpose [2]**
 - Allows the network to be divided into smaller networks
 - ... which reduces traffic in some parts of the network // reduces congestion

- ... because traffic only travels through the parts necessary
- ... which hides the complexity of the network
- **Parts on IP address in subnetwork [2]**
 - Each device in a subnetwork has the same network ID // Each subnetwork has a different network ID
 - Every device in each subnetwork has a different host ID but the same network ID // the host ID uniquely identifies the device within the subnetwork
- **Find network ID and host ID [2]**
 - network ID is when subnet mask is 255
 - host ID is when subnet mask is 0

A subnet mask is used when subnetting a LAN.

Two devices on the LAN are located in different subnetworks.

The IP addresses and corresponding subnet masks are shown:

Device IP address	Subnet mask
10.10.12.1	255.0.0.0
192.168.12.4	255.255.255.0

Identify the following network ID and host ID.

The network ID for the device with the IP address 10.10.12.1

10

The host ID for the device with the IP address 192.168.12.4

4

[2]

- e.g.
- **Benefit [4]**
 - Reduce amount of traffic in a network // improve network speed
 - ... by localising network communications // by dividing data between segments
 - ... so that devices are not flooded with data

- Data stays in its subnet so it does not travel as far
- Improves network security
- ... since not all devices can access all areas of the network
- ... since compromised device does not expose the whole network
- Allows easier maintenance
- ... because only on subnetwork may need taking down/ changing while the rest of the network can continue
- Make network easier to expand
- ... by allowing for greater range of IP addresses to be available
-
- **Public IP address**
 - **purpose [1]**
 - so the device is visible and accessible by other devices on the internet
- **Private IP address**
 - **purpose [1]**
 - so the device is visible and accessible within a LAN
 - **Justify choosing [2]**
 - Improved security because the IP address is not visible outside the network
 - An internet presence is not required for each computer
 - Only the router needs a public IP address, as only the router needs to be externally visible

- Reduces number of (public) IP addresses needed

- **Difference between public and private IP addresses [2]**
 - Public address can be reached across the Internet.
 - Private address can only be reached internally/through the LAN/intranet // private address cannot be reached across the Internet
 - NAT (Network Address Translation) is necessary for a private IP address to access the Internet directly
 - A private address is more secure than a public address // a public address is less secure than a private address
 - Public addresses are provided by ISP / assigned by InterNIC // Private addresses are assigned by the router (of the network concerned)
 - Public addresses are unique (to the Internet) // Private addresses (are unique within their network, but) can be duplicated within other (discrete) networks.
 - 10.0.0.1 to 10.255.255.254 and 172.16.0.1 to 172.31.255.254 and 192.168.0.1 to 192.168.255.254 from the private address space // IP addresses from the private address space are never assigned as public.

- **Static IP address [1]**
 - an IP address that is fixed / doesn't change each time a device re-joins a LAN / network

- **Dynamic IP address [1]**
 - an IP address that may be refreshed / changed each time a device re-joins a LAN / network

- **Why web server use static IP address [3]**

- Static IP does not change
- ... the DNS does not need updating
- ... which might be delayed causing 'address not found' errors
- The web server may be accessed directly using just the IP address // the IP address is still held in cache memory

2.1.15 Explain how a Uniform Resource Locator (URL) is used to locate a resource on the World Wide Web (WWW) and the role of the Domain Name Service (DNS) [4]

- URL is a reference address to a resource on the Internet
- The URL is passed to the nearest Domain Name Server (by browser software)
- DNS server stores a database / list of URLs and matching IP addresses.
- DNS looks for the URL in its database
- Finds the matching IP address and returns it to the originator
- If not found, URL forwarded to another DNS at higher level
- (Original) DNS server adds the returned IP address to its cache
- (Original) DNS server returns the IP address to the browser
- **Further:**
 - Browser uses IP address to request the required web page from the **web server**.
 - Web server retrieves the page and delivers it to the originator
 - Browser software interprets the script(if any) and display the web page

3 Hardware

3.1 Computers and their Components

3.1.1 Show understanding of the need for input, output, primary memory and secondary (including removable) storage

- **Need of secondary storage device [2]**
 - Additional secondary file storage // storing files
 - Backup files
 - Archiving files
 - Transfer files to second computer

3.1.2 Show understanding of embedded systems

Including: [benefits and drawbacks of embedded systems](#)

- **Embedded system**
 - **Define [1]**
 - Microprocessor/microcontroller within a larger system // microprocessor/microcontroller that performs one specific task
 - **Describe [3]**
 - Built into a larger system
 - Dedicated to a single task
 - Function of the system not easily changed by the owner // **Dedicated hardware**
 - **Drawback [1]**
 - Difficult to change / update firmware by the user // Difficult to upgrade devices to take advantage of new technology

- Cannot be easily adapted for another task
- Troubleshooting faults/upgrading is a specialist task
- Devices are often thrown away than repaired as difficult to upgrade or repair
- **Example [1]**
 - Embedded system in washing machine only controls the programs for the washing cycle // it is part of the washing machine but does not perform any other function within it

3.1.3 Describe the principal operations of hardware devices

Including: Laser printer, 3D printer, microphone, speakers, magnetic hard disk, solid state (flash) memory, optical disc reader/writer, touchscreen, virtual reality headset

- **Keyboard [2] (no longer in syllabus)**
 - Uses switches and circuits to translate keystrokes into signals the computer can understand
 - The key matrix is a grid of circuits / three layers of plastic underneath the keys
 - Each circuit is broken beneath the key / middle layer contain holes
 - When key pressed, a circuit is made / completed and a signal is sent
 - Processor compares location of signal from key matrix to a character map stored on ROM
 - A character code for each key press is saved in a keyboard buffer
- **Scanner [2] (no longer in syllabus)**
 - Main component of a scanner is a CCD array

- CCD is a collection of light sensitive diodes
- Laser beam / light is shone onto the source document/
barcode
- The scanned image reaches the CCD through mirrors and
lenses
- Sensors detect levels of reflected light
- Brighter light results in greater electrical charge
- Light intensity is converted (by software) to a digital value
- **Inkjet printer** *(no longer in syllabus)*
 - **Print head [5]**
 - Print head contains a large number of very small nozzles
 - Ink is fed to each nozzle from a reservoir
 - The print head fires droplets of ink onto the paper
 - The print head moves horizontally across the paper
 - Either:
 - Tiny resistors create heat inside each nozzle
 - The heat vaporises ink to create a bubble
 - When the bubble pops the ink is deposited on the page
 - The collapsing bubble creates a partial vacuum in the
nozzle
 - And ink is drawn from the reservoir ready for printing the
next dot
 - Or:
 - There is a piezo crystal at the back of the ink reservoir of
each nozzle
 - The crystal vibrates when it receives a tiny electric
charge
 - Ink is forced out of the nozzle by the inward vibration

- The outward vibration creates a partial vacuum in the nozzle
- Replacement ink is pulled into the reservoir
- **Stepper motor [2]**
 - The (print head) stepper motor is connected to the print head by a belt
 - The (print head) stepper motor moves the print head across the paper
 - The (parking) stepper motor parks the print head assembly when not in use
- **Laser printer [3]**
 - Laser beam and rotating mirror are used to draw an image of the page on the photosensitive drum
 - Image is converted on the drum into an electrostatic charge
 - Electrostatic charge attracts toner
 - The charged paper is rolled against the drum
 - The oppositely-charged paper picks up the toner particles from the drum. After picking up the toner, the paper is discharged to stop it clinging to the drum
 - The electrical charge is removed from the drum and the excess toner is collected.
- **3D printer**
 - **Operation [3]**
 - Additive manufacturing
 - Use a digital 3D model or a Computer Aided Design (CAD) (file)
 - Build up model one layer at a time
 - ... starting from the bottom
 - ... using x, y, z co-ordinates

- Material is heated and pushed through nozzle
- **Purpose of temperature sensor [2]**
 - To prevent overheating // ensure material is hot enough
 - ... by identifying the temperature of the object (being printed)
 - ... by identifying the temperature of the material being used
- **Microphone [3]**
 - The microphone has a diaphragm
 - The incoming sound waves cause vibrations
 - ... causing a coil to move past a magnet (dynamic microphone) // changing the capacitance(condenser microphone)
 - An electric current is generated / changed
- **Speaker [4]**
 - Takes an electrical signal and translate it into physical vibrations to create sound waves
 - An electric current in the coil creates an electro-magnetic field
 - Changes in the audio signal causes the direction of the electric current to change
 - The direction of the current determines the polarity of the electro-magnet // changing the direction of the current changes the direction of the polarity of the electro-magnet
 - The electro-magnet is repelled by or attracted to the permanent magnet
 - Causing the coil to vibrate
 - The movement of the coil causes the cone / diaphragm to vibrate

- That vibration is transmitted to the air in front of the cone / diaphragm as sound waves
- The amount of movement will determine the frequency and amplitude of the sound wave produced
- **Hard disk drive (HDD)**
 - **Operation [4]**
 - The hard disk has one or more platters made of aluminium glass
 - Each surface of the platter/disk is ferrous-oxide which is capable of being magnetised
 - The platters/disks are mounted on a central spindle
 - The disks are rotated at high speed
 - Each surface of the disk has a read/write head mounted on an arm positioned just above the surface
 - Electronic circuits control the movement of the arm and hence the heads
 - The surface of the platter/disk is divided into concentric tracks and sectors
 - One track in one sector is the basic unit of storage called a block
 - The data is encoded as a magnetic pattern for each block
 - When writing to disk, a variation in the current in the head produces a variation in magnetic field on the disk
 - When reading from the disk, a variation in magnetic field produces a variation in current through the head
 - **Advantage to SSD [1]**
 - Inexpensive per unit storage
 - Larger storage capacity than flash drive

- **Justify choice [2]**
 - Large capacity
 - Reasonably fast access speed
 - Inexpensive per unit storage
 - Does not need to be moved (according to question)
 - Slower degradation of data, last longer
- **Flash memory (SSD)**
 - **Describe [1~3]**
 - NAND-based flash memory
 - No moving parts
 - Uses a grid of columns and rows that has two transistors at each intersection
 - One transistor is called a floating gate
 - Second transistor called the control gate
 - Memory cells store voltages which can represent 0 and 1
 - Movement of electrons is controlled to read/write
 - Not possible to over-write existing data, have to first erase the old data.
 - **Advantage to HDD [1]**
 - No moving parts / noise
 - Robust
 - Low latency // Fast read/write time
 - **Justify choice [2]**
 - Large capacity
 - Fast access speed
 - Reliable (*since no moving parts*)
 - Quiet (*since no moving parts*)
- **Optical disc [2]**

- The disc is spun at high speed
- A laser is shone onto the disc to read / write
- ... using optical head to move it into position
- ... it follows the spiral track from the centre outwards
- When writing the laser burns pits to represent the data
- When reading the laser reflects from pits and lands
- The reflection from a pit and a land is different
- ... the differences are interpreted as 1 or 0
- **Optical mouse (no longer in syllabus) [2]**
 - Laser / light shines onto a surface
 - Through a (polished) ring at the base
 - The light is reflected from the surface through the ring
 - Sensor detects reflected light
 - Capturing details / photograph of surface
 - At about 1500 times per second
 - As the mouse moves the sensor detects changes in the surface detail / photograph
 - Which are translated into movement (change of x and y co-ordinates)
 - The processor/software updates the position of the cursor on the screen
- **DVD-RAM [1]**
 - data stored/written using laser
 - uses phase changing recording, in which varying laser intensities cause targeted area in the phase change recording layer to alternate between an amorphous and a crystalline state
 - uses a rotating disk with concentric tracks
 - allows read and write operation to occur simultaneously

- **DVD-RW and CD-RW** (*analysed from textbook*)
 - data stored/written using laser
 - When writing laser burn surface metal to liquid form, which is then changed into either a crystalline or an amorphous solid depending on heat of laser
 - crystalline and amorphous reflect laser differently, which is encoded into 0 and 1
 - uses a rotating disk with spiral tracks
 - read and write operation must be done separately
- **Touch screen**
 - **Resistive [4]**
 - Consist of two charged plates
 - Pressure causes the plates to touch
 - Completing the circuit
 - Point of contact identified
 - Coordinates used to calculate the position
 - **Capacitive [4]**
 - Made from materials that store electric charge
 - When touched charge transferred to the finger
 - Sensors at the corners detect the change
 - Point of contact identified
 - Coordinates used to calculate the position
- **VR headset**
 - **Operation [4]**
 - Have one or two screens that output image to the user
 - Has speakers that output surround sound to give realistic experience
 - User's movements detected using an accelerometer.

- Data transmitted to a microprocessor that identifies the direction and speed of movement
- Some headsets use digital camera that record user's eye movements for analysis

3.1.4 Show understanding of the use of buffers

- Buffer

- Purpose [2]

- to act as temporary storage // to store (downloaded) data
- ... before it is used by the receiving device
- ... to allow processes / devices to operate at different speeds // independently of each other

- Operation (transfer data to external device) [3]

- Computer and device transmit and receive at different speeds
- Act as a temporary store for data going to device
- Data is transferred into buffer by computer
- ... and is not held up waiting for data to transfer
- Data is retrieved from buffer by device at slower rate
- When buffer is empty/full an interrupt is sent to the computer requesting more data/stopping data transfer

- Example usage [1]

- printer buffer used when data is transferred from a computer to a printer
- video buffer when streaming videos
- keyboard buffer when performing data entry

- Describe how it's possible to watch video without pausing online [4]

- Use needs high-speed broadband.
- Data is streamed to a buffer (in the computer)
- Buffering stops video from pausing when streaming
- As buffer is emptied, it fills up again so that watching is continuous
- Actual playback is behind the time the data is received by computer.

3.1.5 Explain the differences between Random Access Memory (RAM) and Read Only Memory (ROM) [2]

Including their use in a range of devices and systems

- RAM

- loses contents when power turned off/ volatile memory / temporary memory
- stores files/data/operating system currently in use.
- data can be changed
- memory size often larger than ROM

- ROM

- doesn't lose content when power turned off / non-volatile memory / permanent memory
- can't be changed
- can be used to store BIOS / bootstrap / pre-set instructions

3.1.6 Explain the differences between Static RAM (SRAM) and Dynamic RAM (DRAM) [3]

Including the use of SRAM and DRAM in a range of devices and systems and the reasons for using one instead of the other depending on the device and its use.

- DRAM has to be refreshed / charged
// SRAM does not request a refresh
- DRAM uses a single transistor and capacitor
// SRAM uses more than one transistor to form a memory cell
// SRAM has more complex circuitry
- DRAM stores each bit as a charge
// SRAM each bit is stored using a flip-flop/latch
- DRAM less expensive (to purchase / since it requires fewer transistors)
// SRAM is more expensive
- DRAM has slower access time / speed (because it needs to be refreshed)
// SRAM has faster access times
- DRAM can have higher storage / bit / data density
// SRAM has lower storage / bit / data density
- DRAM used in main memory
// SRAM used in cache memory
- **DRAM advantage [2]**
 - Cost less per unit
 - higher bit density per chip
 - Simple design - use fewer transistors

3.1.7 Explain the difference between Programmable ROM (PROM), Erasable Programmable ROM (EPROM) and Electrically Erasable Programmable ROM (EEPROM)

- **Explain difference between them [3]**

- PROM can be set once, and EPROM and EEPROM can be overwritten multiple times
- EPROM needs to be removed from the device for erasing, while EEPROM can be erased in situ.
- EPROM can be erased using UV light, EEPROM can be erased using voltage // is flash storage
- EPROM must be entirely erased before writing, EEPROM does not have to be entirely erased before rewriting

- **EEPROM benefit**

- no additional equipment needed to change
- ... enable firmware update by non-technical users
- can be erased and reprogrammed several times
- ... so firmware can be updated
- ... can erase a particular byte or the whole EEPROM // without fully erasing the contents of the firmware in the headset first
- possible to reprogram / update
- ... without removing it from the device

3.1.8 Show an understanding of monitoring and control systems

Including:

- difference between monitoring and control system
- use of sensors (including temperature, pressure, infra-red, sound) and actuators
- importance of feedback

- **Monitoring system**

- **Operation [4]**

- sensors send signals / data to a computer
- data / signal converted to digital (using an ADC)
- (computer processes the signal)
- if ... (value is greater / smaller than ...)
- then ... (send signal to)
- monitoring is continuous

- **Control system**

- **Characteristics [3]**

- Uses feedback
- Acts independently
- Allow output to influence input
- **Include actuator / produce action**

- **Actuator [2]**

- Actuator generates a signal / causes an action / convert electrical energy into a mechanical force
- ... to push an arm, open a trap door, etc.

- **Difference between control & monitor [2]**

- control system uses feedback

- control system produce action
- control system have actuator
- control system produces action to **autonomously** change the environment if the values are out of prescribed range
- output from a control system affects the next input
- *(this really depends on the preference of the examiner)*
- **Types of sensors**
 - temperature
 - pressure
 - infra-red
 - light
 - sound/acoustic
 - accelerometer
- **Importance of Feedback [2]**
 - Feedback ensures that a system operates within set criteria / constraints
 - ... by enabling system to output to affect the system input
 - ... thus allowing conditions to be automatically adjusted

3.2 Logic Gates and Logic Circuits

3.2.1 Use the following logic gate symbols:



NOT



AND



OR



NAND



NOR



XOR

3.2.2 Understand and define the functions of : NOT, AND, OR, NAND, NOR and XOR (EOR) gates

All gates except the NOT gate will have two inputs only.

- OR [1]

- The output is 0 when both inputs are 0, otherwise the output is 1

- NAND [1]

- 0 is only output when both inputs are 1 // 1 is only output when none, or (either) one of the inputs is 1

- NOR [1]

- 1 is only output when both inputs are 0 // 0 is only output when (either) one or both inputs are 1

- XOR [1]

- The output is 1 when one of the inputs is 1 and the other input is 0, otherwise the output is 0

3.2.3 Construct the truth table for each of the logic gates above

3.2.4 Construct a logic circuit

From:

- a problem statement
- a logic expression
- a truth table

3.2.5 Construct a truth table

From:

- a problem statement
- a logic expression
- a truth table

3.2.6 Construct a logic expression

From:

- a problem statement
- a logic expression
- a truth table

- Method to construct a logic expression

- Extract conditions that output 1

- Example:

(b) Write the logic expression for the following truth table.

R	S	T	Q
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

- Q = ... [2]

1 mark per bullet point, max 2 marks

- One correct term: (R AND S AND NOT T)
// (NOT R AND NOT S AND T)
- Second correct term **plus** the OR in correct place

Answer:

- Q = (R AND S AND NOT T) OR (NOT R AND NOT S AND T)

4 Processor Fundamentals

4.1 Central Processing Unit (CPU) Architecture

4.1.1 Show understanding of the basic Von Neumann model for a computer system and the stored program concept

- **Stored program concept [1]**

- Instructions and data are stored in the same memory space / in main memory

4.1.2 Show understanding of the purpose and role of registers, including the difference between general purpose and special purpose registers

Special purpose registers including:

- Program Counter (PC)
- Memory Data Register (MDR)
- Memory Address Register (MAR)
- The Accumulator (ACC)
- Index Register (IX)
- Current Instruction Register (CIR)
- Status Register

- **What is register [2]**

- Small piece / word of (fast) memory
- Part of the processor
- Temporary storage of data

- Data is about to be / has been processed
- **Difference between special purpose register and general purpose register [2]**
 - Special purpose registers hold the status of a program whereas general purpose registers hold the temporary data while performing operations
 - Special purpose registers are specialised for a specific use, whereas general purpose registers are used for any purpose
 - General purpose registers can be used by most instructions, whereas special purpose registers can only be used by certain instruction
- **The Accumulator [1]**
 - stores the intermediate results of arithmetic and logical operations // holds the result of a calculation
- **Program counter [1~2]**
 - stores the address of next instruction to be executed
- **Memory Data Register [2]**
 - stores the data in transit between memory and other registers
 - holds the instruction before it is passed to the CIR
 - Stores / holds data / instruction when fetched from memory
 - Stores / holds data which is being written to memory
 - The location accessed is the address held in the Memory Address Register (MAR)
- **Memory Address Register [1]**
 - stores the address of the memory location currently being read from or written to (*remember to include written to*)
- **Index register [1]**

- to store a value that is added to an operand to give another address
- **Current Instruction Register [1]**
 - stores the current instruction being executed
- **Status Register [2]**
 - Interpreted as independent bits / flags
 - Each flag is set depending on an event / after arithmetic / logical operation
 - e.g. addition overflow / result of operation is zero etc.

4.1.3 Show understanding of the purpose and roles of the Arithmetic and Logic Unit (ALU), Control Unit (CU) and system clock, Immediate Access Store (IAS)

- **ALU [2]**
 - performs arithmetic operations
 - And logical operations / comparisons
- **Control Unit [2]**
 - send / receive control signals along control bus
 - synchronises operations of other components in the CPU
 - to manage to execution of instructions (in sequences)
 - to control the communication between the components of the CPU
- **System clock [2]**
 - synchronise operations
 - ... by creating timing signals
 - to keep track of the date and time / timestamp file
 - to process operations in the correct order / sequence
- **Immediate Access Store (IAS) [2]**

- Holds all data/instructions/programs currently in use
- Is volatile memory
- Has fast access times

4.1.4 Show understanding of how data are transferred between various components of the computer system using the address bus, data bus and control bus

- Address bus [1]

- lines used to transfer address of memory or input/output location
- unidirectional bus

- Data bus [1]

- used to transfer data between the processor and memory/ input and output devices
- bidirectional bus

- Control bus

- Describe [1]

- used to transmit control signals
- e.g. read/write/fetch
- dedicated bus since all timing signals are generated according to control signal

- Example signals [2]

- Interrupt
- Timing
- Read
- Write

- **Explain how control bus, system clock, the CU operate to transfer data between the components of the computer system [2]**
 - The system clock gives out timing signals
 - ... which are sent on the control bus
 - ...to synchronise the other system components
 - The Control Unit initiates data transfer
 - ...by generating signals that are sent on the control bus to other components

4.1.5 Show understanding of how factors contribute to the performance of the computer system

Including:

- processor type and number of cores
- the bus width
- clock speed
- cache memory
- **Number of cores [2]**
 - Each core processes one instruction per clock pulse
 - More/multiple cores mean that sequences of instructions can be split between them
 - and so more than one instruction is executed per clock pulse // more sequences of instructions can be run at the same time // **more instructions are performed in parallel**
 - More cores decrease the time taken to complete task
- **Why two cores may not run two times faster than one [5]**
 - Multiple cores introduce additional overheads
 - **Latency may be increased**

- ... because of the need for communication between cores
- Software may not be designed for multiple cores...
- ... so one of the cores will be left aside
- Memory access speed may not match speed of cores...
- ... so causing delay
- May have more differences than just cores
- ... more RAM allows faster multitasking
- ... may have GPU
- ... etc.
- **Data bus width [2]**
 - the width of the data bus determines the number of bits that can be simultaneously transferred
 - increasing the width of the data bus increase the number of bits can be moved at one time
 - ... hence improving the processing speed as fewer transfers are needed
 - By example: e.g. double the width of the data bus moves 2x data per clock pulse.
- **Address bus width [1]**
 - Increase in address bus width increase the number of directly addressed memory locations
- **Clock speed [2]**
 - Each instruction is executed on a clock pulse // one F-E cycle is run on each clock pulse
 - ... so clock speed determines the number of instructions that can be executed per second.
 - Faster clock speed, more instructions can be run per second.

- ... however, there is a limit on clock speed because the heat generated by higher clock speeds can't be removed fast enough. *(appeared once)*

- Cache [2]

- higher the capacity of cache, more frequently used instructions it can store for fast access *(if one mark)*
- cache is fast access memory (close to CPU)
- cache stores frequently used instruction / data
- ... more cache means more instructions / data can be transferred faster
- ... less swapping between RAM and cache
- prevents CPU idling while waiting for data

4.1.6 Understand how different ports provide connection to peripheral devices

Including connection to:

- Universal Serial Bus (USB)
- High Definition Multimedia Interface (HDMI)
- Video Graphics Array (VGA)

- Purpose of port [1]

- To provide connection to peripheral devices
- To provide an interface between the computer and other devices

- USB

- Operation [1]

- 1 bit is transferred at a time
- Can be synchronous or asynchronous
- USB-3 is full-duplex and earlier versions are half-duplex

- Automatic connection [2]

- A voltage change occurs when the device is plugged in
- The computer detects this voltage change
- The code of the device is transferred to computer
- ... the OS finds the code of the device in the list of devices
- ... and load the appropriate driver
- **Benefit [4]**
 - Automatic connection
 - ... so usually no need to install separate device driver
 - USB has become an industrial standard
 - ... so most computers will be equipped with USB ports
 - Devices may be powered or charged through USB
 - ... so external devices may be charged while working at the computer
 - nearly impossible to wrongly connect a device
 - USB 3.0 allows full duplex data transfer
 - backwards compatible
- **Justify choosing [2]**
 - Has fast data transfer speed
 - Is a universal/popular cable // universal standard
- **HDMI**
 - **Benefit (to VGA) [4]**
 - HDMI has faster transfer rates than VGA
 - ... needed due to high resolution / large number of pixels of monitor // HDMI supports the high resolution of the monitor
 - HDMI supports video and audio transfer between computer and monitor speakers
 - ... so no separate sound cable is needed unlike VGA

- HDMI is digital interface therefore no data is lost in transfer to analogue and back
- HDMI is less prone to error/crosstalk/external interference
- **Justify choosing [2]**
 - Allows video and audio to be transferred on the same cable
 - convenience as no need for two cables
- **VGA** (*analysed from textbook*)
 - does not transmit the audio component
 - (*from the HDMI question*) data need to be transferred to analogue and back

4.1.7 Describe the stages of the Fetch-Execute (F-E) cycle

Describe and use 'register transfer' notation to describe the F-E cycle

- **Fetch stage [4]**
 - Program Counter(PC) holds the address of the next instruction to be fetched
 - The address in the PC is copied to the Memory Address Register(MAR)
 - The (*content in*) PC is incremented
 - The instruction is copied to the Memory Data Register(MDR)
 - ... from the address held in the MAR
 - ...which has been read / written
 - The instruction from the MDR copied to Current Instruction Register
- **Register transfer notation [3]**
 - **Use [3]**

```
MAR <- [PC]
PC <- [PC] + 1
MDR <- [ [MAR] ]
CIR <- [MDR]
```

- **Explain [[MAR]] [1]**

- The contents from the MAR is an address, it is the contents of that address which is transferred to MDR
- The contents of the address pointed to by the MAR is transferred to the MDR

4.1.8 Show understanding of the purpose of interrupts

Including:

- possible causes of interrupts
- applications of interrupts
- use of an Interrupt Service handling Routine (ISR)
- when interrupts are detected during the fetch-execute cycle
- how interrupts are handled

- **Purpose / Definition [2]**

- to send a signal from a device or process
- ... seeking the attention of the processor

- **Operation [4~5]**

- at the end of the fetch-execute cycle ...
- check for interrupt // if an interrupt flag is set / bit set in interrupt register
- Priority checked
- If lower priority than current process continue with F-E cycle
- If higher priority than current process...
- ... all contents of registers are saved to stack
- PC loaded with address of interrupt service routine.

- When ISR finished, check for further interrupts
- if no more interrupts of higher priority, load data from stack and continue with process
- **Possible causes [3]**
 - Hardware fault
 - I/O request
 - Program / software
 - division by zero // runtime error in a program
 - attempt to access an invalid memory location
 - array index out of bounds
 - stack overflow
 - End of a time-slice

4.2 Assembly Language

4.2.1 Show understanding of the relationship between assembly language and machine code

(analysed from textbook)

- If a program has been written in assembly language it has to be translated into machine code before it can be executed by the processor.
- translation program is called an assembler

4.2.2 Describe the different stages of the assembly process for a two-pass assembler

Apply the two-pass assembler process to a given simple assembly language program

- **First pass**

- **Purpose [1]**
 - To create a symbol table
- **Describe [2]**
 - Read the assembly language program one line at a time
 - Remove comments
 - Check the opcode is in the instruction set // To create a symbol table
- **Process of symbol table creation [3]**
 - The assembler scans the assembly language instructions in sequence
 - When it meets a symbolic address it checks to see if already in symbol table
 - If not, it adds it to the symbol table in the symbolic address column
 - If it is already in symbol table check if absolute address known
 - If the absolute address is known, it is entered in the appropriate cell
 - If the absolute address is not known leave as unknown
- **Explain directive [2] (not directly in syllabus)**
 - An instruction that directs the assembler to do something
 - A directive is not a program instruction
 - It is information for the assembler
 - By example (State the start address of the program // tell the assembler to set aside space for variables // include external files etc.)
- **Explain Macro [2] (not directly in syllabus)**
 - A group of instructions given a name // subroutine

- A group of instructions that need to be executed several times within the same program
- The statements are written once and called using the name whenever they need to be executed
- Macro code is inserted into the source file at each place it is called
- By example
- **Second pass**
 - Read the assembly language program one line at a time
 - Generate the object code
 - Jump instructions access memory addresses via table
 - save or execute the program, error repeated if they exist

4.2.3 Trace a given simple assembly language program [5~7]

Usually, a table explaining the instructions will be given here. However, it is still important to know that **CMP #n** is in essence testing if **[ACC] == n**. That is, it would return TRUE if and only if ACC holds the exact same value as the operand.

4.2.4 Show understanding that a set of instructions are grouped

Including the following groups:

- Data movement
- Input and output of data
- Arithmetic operations
- Unconditional and conditional instructions
- Compare instructions

- **Data movement** (*analysed from textbook*)
 - loading data into a register or storing data in memory.
- **Input and output data [1]**
 - Takes an input from the user // outputs the character of the binary number
- **Arithmetic operations [1]**
 - Perform addition and subtraction
- **Unconditional and conditional instructions [1]**
 - Move to another instruction (identified by a label)
- **Compare instructions [1]**
 - Compare the result to another value

Data movement	Input and output	Arithmetic	Unconditional and conditional insturction	Compare
LDD	IN	ADD	JPE	CMP
LDX	OUT	SUB	JPN	CMI
LDR		INC	JMP	
LDI		DEC		
LDM				
MOV				
STO				

4.2.5 Show understanding of and be able to use different modes of addressing

Including immediate, direct, indirect, indexed, relative

- Immediate addressing [1]

- the operand is the value to be used

- Direct addressing [1]

- the operand is the address of the value to be used
- LDD #operand:
 - the MAR is loaded with the operand of the instruction
 - Accumulator loaded with the contents of the address held in MAR.

- Indirect addressing [1]

- the operand is the address of the address of the value to be used

- Indexed addressing [1]

- the operand plus contents of index register is the address of the value to be used

- **Relative addressing**
 - **Describe [1]**
 - the operand is the offset from the current address where the value to be used is stored.
 - **Purpose [2]**
 - To allow for re-locatable code
 - ... because all (target) addresses can be specified by the base address + offset
- **Absolute addressing [1]** (no longer in syllabus)
 - The operand is a numeric address // The numeric address is given // referring directly to a memory location
- **Symbolic addressing [1]** (no longer in syllabus)
 - The operand is a word/symbol // A word/symbol represent the memory location/address

4.2.6 An example instruction set

Instruction		Explanation
Opcode	Operand	
LDM	#n	Immediate addressing. Load the number n to ACC
LDD	<address>	Direct addressing. Load the contents of the location at the given address to ACC
LDI	<address>	Indirect addressing. The address to be used is at the given address. Load the contents of this second address to ACC
LDX	<address>	Indexed addressing. Form the address from <address> + the contents of the index register. Copy the contents of this calculated address to ACC
LDR	#n	Immediate addressing. Load the number n to IX
MOV	<register>	Move the contents of the accumulator to the given register (IX)
STO	<address>	Store the contents of ACC at the given address
ADD	<address>	Add the contents of the given address to the ACC
ADD	#n/Bn/&n	Add the number n to the ACC
SUB	<address>	Subtract the contents of the given address from the ACC
SUB	#n/Bn/&n	Subtract the number n from the ACC
INC	<register>	Add 1 to the contents of the register (ACC or IX)
DEC	<register>	Subtract 1 from the contents of the register (ACC or IX)
JMP	<address>	Jump to the given address
CMP	<address>	Compare the contents of ACC with the contents of <address>
CMP	#n	Compare the contents of ACC with number n
CMI	<address>	Indirect addressing. The address to be used is at the given address. Compare the contents of ACC with the contents of this second address
JPE	<address>	Following a compare instruction, jump to <address> if the compare was True
JPN	<address>	Following a compare instruction, jump to <address> if the compare was False
IN		Key in a character and store its ASCII value in ACC
OUT		Output to the screen the character whose ASCII value is stored in ACC
END		Return control to the operating system

All questions will assume there is only one general purpose register available (Accumulator)
 ACC denotes Accumulator
 IX denotes Index Register
 <address> can be an absolute or symbolic address
 # denotes a denary number, e.g. #123
 B denotes a binary number, e.g. B01001010
 & denotes a hexadecimal number, e.g. &4A

4.3 Bit Manipulation

4.3.1 Show understanding of and perform binary shifts

Logical arithmetic and cyclic

Left shift, right shift

- **Logical shift [1]**

- Moves all bits to one direction and insert zero in the other end
- e.g. two's complement number 11001010 after a logical left shift of 2 places
- 0010 1000

- **Right arithmetic binary shift [1]**

- Moves all bits to the right but copies the sign bit into the Most Significant Bit (MSB)
- e.g. two's complement number 10011110 after arithmetic right shift of 3 places
- 1111 0011
- e.g. Identify the mathematical operation of LSR #3
- The number is divided by 8 (and only whole number retained)

- **Cyclic** (*analysed from textbook*)

- Bits moves off one end into the carry bit then one step later moves in at the other end.
- All bit values in the original code are retained.

4.3.2 Show understanding of how bit manipulation can be used to monitor/control a device

- **Test whether number in ACC is odd [3]**

- An odd binary number will have a 1 in the Least Significant Bit (LSB)
- A bit manipulation operation is required to access/mask only the LSB and clear all the others
- Compare the result of the masking with denary 1
- ... the result of comparison will be true if the number is odd
- AND B00000001 // AND #1 // AND &1
- **Clear data in a 8 bit register [3]**
 - A bit manipulation is required to set all bits to zero
 - Compare the result of the masking with 0
 - ... the result of the comparison will be true if the register is cleared
 - AND B00000000 / #00 / &00

Label	Instruction		Explanation
	Opcode	Operand	
	AND	#n / Bn / &n	Bitwise AND operation of the contents of ACC with the operand
	AND	<address>	Bitwise AND operation of the contents of ACC with the contents of <address>
	XOR	#n / Bn / &n	Bitwise XOR operation of the contents of ACC with the operand
	XOR	<address>	Bitwise XOR operation of the contents of ACC with the contents of <address>
	OR	#n / Bn / &n	Bitwise OR operation of the contents of ACC with the operand
	OR	<address>	Bitwise OR operation of the contents of ACC with the contents of <address>
	LSL	#n	Bits in ACC are shifted logically n places to the left. Zeros are introduced on the right hand end
	LSR	#n	Bits in ACC are shifted logically n places to the right. Zeros are introduced on the left hand end
<label>:	<opcode>	<operand>	Labels an instruction
<label>:		<data>	Gives a symbolic address <label> to the memory location with contents <data>

All questions will assume there is only one general purpose register available (Accumulator)

ACC denotes Accumulator

IX denotes Index Register

<address> can be an absolute or symbolic address

denotes a denary number, e.g. #123

B denotes a binary number, e.g. B01001010

& denotes a hexadecimal number, e.g. &4A

5 System Software

5.1 Operating System

5.1.1 Explain why a computer system requires an Operating System (OS) [2]

- The hardware is unusable without an OS // hides complexity of hardware from user
- Act as an interface / controls communications between user and hardware / hardware and software
- Provides software platform / environment on which other programs can be run

5.1.2 Explain the key management tasks carried out by the Operating System [4~6]

Including memory management, file management, security management, hardware management (input / output / peripherals), process management

- **Main memory management**
 - Memory protection to ensure that two programs do not try to use the same space
 - Use of virtual memory
 - Deciding which processes need to be in the main memory at any one time
 - Location of processes within the memory
 - **Paging // segmentation**
- **File / secondary storage management [4]**
 - Maintains directory structures
 - Provide file naming conventions

- Control access to the file
- Specific tasks that can be performed on a file (e.g. open, close, delete, copy, etc.)
- Space is allocated to particular files
- Storage space is divided into file allocation units
- Specifies the logical method of file storage (e.g. FAT or NTFS)
- **Security management**
 - Makes provision for recovery when data is lost / Automatic backup
 - Provides usernames and passwords / encryption / user accounts
 - Prevent unauthorised access
 - Ensures privacy of data
 - System restore / roll back (to previous state)
 - Implement access rights
- **Peripheral / hardware / device management [4]**
 - Install device drivers
 - ... to allow communication between peripherals and computer
 - Sends data and receives data to and from peripherals
 - ... such as to an output device and from an input device
 - Handles buffers for transfer of data
 - ... to ensure smooth transfer between devices that transmit and receive at different speeds
 - Manage hardware interrupts / signals from the device
- **Process / resource management [3]**
 - Scheduling of processes
 - allows multi-tasking

- ensures fair access
- handles priorities
- Resolution of conflicts when two or more processes require the same resource
- Manages the resources the processes need
- Enable processes to share information (*rarely come up*)
- **How process management and memory management support multi-tasking [6]**
 - Process management [3]
 - Schedule processes
 - Decide which process to be run next
 - Allow one process to be paused whilst another process can be actioned
 - Switch between processes to allow them to share the use of
 - Memory management [3]
 - store data from currently running program to RAM
 - decide which process should be in main memory
 - make efficient use of memory
 - Stops the data from overwriting each other in RAM
 - *Note : basically there are three common points:*
 - *prevent process using the same memory // overwriting each other in RAM // require same resource*
 - *schedule / decide which process to be run next (process)*
 - *decide which process should be in main memory (memory)*
- **Interrupt handling (no longer in syllabus)**
 - Identifies priorities of interrupts

- save current memory / process values / saves data on power outage
- Loads appropriate Interrupt Service Routine (ISR)

5.1.3 Show understanding of the need for typical utility software provided with an Operating System

Including disk formatter, virus checker, defragmentation software, disk contents analysis/ disk repair software, file compression, back-up software

- Purpose of utility software [2]

- To help users to set-up / configure / analyse / optimise / maintain the computer ...
- ... by for example, making memory allocation is more efficient
- ... by for example, checking the system for faults (*I don't really know what 'by for example' is*)

- Hard disk formatter[1~2]

- Prepares a hard disk to allow data to be stored on it (if one mark)
- Makes existing data inaccessible (*not needed if new disk*)
- Sets up the (specified) file system
- Prepares the disk for initial use
- May check for errors on the disk
- Partitions the disk into logical drives

- Set up file allocation table
- **Virus checker [2]**
 - Scans files stored on a computer system for malicious code
 - Scans files when they enter the system / memory stick inserted / downloaded etc.
 - Set up a schedule of virus-checking
 - Isolates / quarantines / deletes viruses
 - Regularly updates the virus definitions
- **Hard disk defragmenter [2]**
 - **Describe [2]**
 - Re-organises the disk contents
 - Moves split files so they are contiguous
 - Creates a larger area of (contiguous) free space
 - **Why improve performance of computer [3]**
 - Rearrange blocks of individual files so they are contiguous // move free space together
 - Accessing each file is faster
 - ... because there is no need to search for the next fragment / block of the file
 - ... so less head movement is needed
- **Disk repair software [3]**
 - Checks for any errors / inconsistencies / bad sectors on the disk
 - Resolves any errors on the disk
 - Retrieves files / data from a damaged disk // re-constructs directory // recovers disc when data corrupt
 - Marks bad sectors on the disk // mark bad sectors as unusable
- **Compression software [2]**

- to reduce the size of files
- ... which saves storage and memory space
- ... and reduce transmission time
- **Backup software [1~2]**
 - To make a copy of data at regular time intervals
 - So that if it is lost it can be retrieved
 - Allows the user to decide what is backed up. e.g. all data // all files that have changed since the last backup
 - Allow user to set up an off-site backup
 - May encrypt backup files
- **Firewall [1]**
 - prevent unauthorised access to computer system from external sources

5.1.4 Show understanding of program libraries

Including:

- software under development is often constructed using existing code from program libraries
- the benefits to the developer of software constructed using library files, including Dynamic Link Library (DLL) files
- **Program library**
 - **Describe [2]**
 - Pre-existing / pre-compiled / pre-written functions and routines / modules / code
 - ... can be imported (linked) into the program (without amendment)
 - The functions/routines can be called in the program // To perform common / complex tasks
 - **Benefit (using) [4]**

- The code is already written
- which saves time

- The code will have been used by many people
- So it should be already thoroughly tested // relatively error-free

- The programmer can use, e.g. mathematical / graphical functions, etc. (may not know how to code)
- Can be sure that the function will perform as it should // simplifies the program.

- The code should conform to industry standards
- And therefore contribute towards a more robust program

- **Benefit (creating) [3]**
 - Subroutines can be shared / reused
 - ... between team members who are working independently
 - ... without having to rewrite/re-test them which saves time
 - A program library provides continuity between programs
 - Individual programmers can contribute their specialisms to the library

- **Drawback [2]**
 - Compatibility issues
 - ... may not work with the other code / may require changing program for it to work

- Not guaranteed thorough testing

- ... may be unknown or unexpected bug / virus
- Library routine may not meet exact needs
- ... may give unexpected results // may need editing
- If library routine is changed
- ... there may be unexpected results / errors
- **Dynamic Link Library(DLL)**
 - **Describe DLL [2]**
 - A collection of self-contained (shared library) programs
 - ... that are already compiled
 - Linked to the main program during execution
 - Library program code is separate from the .EXE file
 - Library file only loaded into memory when required at run time
 - A DLL file can be made available to several applications (at the same time)
 - If DLL routine is updated the program that uses it will run the update
 - **Describe DLL file [2]**
 - A DLL file is a shared library file
 - Code is saved separately from the main .EXE files
 - Code is only loaded into main memory when required at run-time
 - The DLL file can be made available to several applications (at the same time)
 - **Benefit of DLL (general) [4]**
 - The executable file is smaller / the executable does not contain all the library routines ...

- ... DLL files are only loaded into memory when required.
- Changes / improvements / error correction to the DLL file code are done independently of the main program...
- ... So there is no need to recompile the main program
- ... All programs using it will benefit
- A single DLL file can be made available to several application programs ...
- ... Saving space in memory / easing the pressure on memory
- **Benefit of DLL (for developer) [4]**
 - maintenance not needed to be done by the programmer
 - ... because the DLL is separate from program
 - no need to recompile the main program when changes are made to DLL
 - ... because changes to the DLL file code are done independently of the main program
 - ... updates will apply to all programs that use the DLL file
 - *(the first four points always occurs)*
- **Disadvantage of DLL [2]**
 - The executable code is not self-contained ...
 - ... the DLL file needed to be included at run-time.
 - Appropriate (linking) software must be available at run-time ...
 - ... to link / include / import the DLL files.

- The DLL file must be present ...
- ... otherwise (unable to find X.dll) errors
- Unexpected changes to the DLL file / corrupted DLL file
- ... could mean the program stops working as expected
- Malicious changes to the DLL file ...
- ... could install a virus on the user's computer / related files could be corrupted

5.2 Language Translators

5.2.1 Show understanding of the need for:

- **Assembler** (*analysed from textbook*)
 - translate assembly language code into machine code so it can be executed by the processor
- **Interpreter [2]**
 - Used during development
 - Debugging is easier
 - ... Because errors are reported as they are found // No need to wait until the end of the process for the error report // to test/debug the partially completed program
 - ... Because errors can be corrected as they are found
- **Compiler [2]**
 - Compiler used when development complete // compiler used when program ready for distribution
 - Produces an executable file
 - ... After the compilation the compiler does not need to be present for the program to run

- ... The program can be given to others without access to code
- ... Final program does not need to be re-compiled each time it is run
- Cross-compilation, the program can be compiled to run on different platforms

5.2.2 Explain the benefits and drawbacks of using either a compiler or interpreter and justify the use of each

- Compiler

- Operation [2]

- Attempts to translate the whole source code
- Creates a separate error report at the end of the translation process
- If translation successful / no errors creates an executable file

- Benefit [2]

- Once translated the compiler software is not needed to run the program
- Compiled code execute faster
- Compiler produces an executable file
- The executable file produced by a compiler can be distributed without users having sight of the source code
- Cross-compilation is possible

- Drawback [2]

- Code can't be changed without recompilation
- Errors can't be identified in real-time

- Cannot easily test specific sections of the source code // cannot easily test unfinished code
- One error may result in other false errors being reported
- The program will not run if there are any errors
- Slower to produce the object code than an interpreter
- Larger amounts of source code take time to compile (*// don't recommend writing this one*)
- **Interpreter**
 - **Operation [2]**
 - Reads each line and translate it and executes it
 - Stops when an error is encountered // display errors where it finds them
 - **Benefit [2]**
 - Easier de-bugging
 - The interpreter stops when error encountered
 - ... so error can be corrected in real time
 - Only one error is displayed at a time
 - .. so fewer errors to correct simultaneously and no dependent errors
 - The interpreter translates a statement then executes it immediately
 - Parts of the program can be tested without all the program code being available
- **Difference between compiler and interpreter [4]**
 - compiler creates and executable file
 - ... the compiled program can be independently distributed
 - compiler reports all error at the end of compilation // interpreter stops when it reaches an error

- ... Interpreter executes each statement immediately after decoding // compiler checks the whole program for errors
- The interpreter software/source code must be present in main memory every time the program is executed // the compiled program does not require compiler/source code to be present
- ... cross-compilation is possible/compile on one hardware platform to run on another

5.2.3 Show awareness that high-level language programs may be partially compiled and partially interpreted, such as Java (console mode)

- **Partial compilation**

- **Why [2]**

- Partially compiled programs can be used on different platforms as they are interpreted when run
 - code is optimised for the CPU as machine code is generated at run time
 - Source code does not need recompiling so more efficient to run

- **Advantage [1]**

- Programs may be interpreted on different platforms

- **Disadvantage [1]**

- Intermediate code / program still needs to be interpreted on the user's computer, which may run slowly
 - Extra CPU resources may be required

- **Describe Machine independent [1]**

- The program code can be translated to run on any processor / platform
- Source code is translated into machine independent intermediate code not machine dependent code
- **Process of Java source code program compilation [2]**
 - Java uses a two-step translation process
 - Java code is partially interpreted - partially compiled
 - Code is translated first into intermediate code / bytecode ...
 - ... using the Java compiler
 - The bytecode is finally interpreted by the Java Virtual Machine

5.2.4 Describe features found in a typical Integrated Development Environment (IDE)

Including:

- for coding, including context-sensitive prompts
- for initial error detection, including dynamic syntax checks
- for presentation, including pretty-print, expand and collapse code blocks
- for debugging, including single stepping, breakpoints, i.e. variables, expressions, report window
- **Coding**
 - Auto-complete
 - **Auto-correct [1]**
 - Correct spelling mistakes so that user has fewer errors to correct
 - **Context sensitive prompts [1~2]**
 - Suggest the code to add as the user types
 - As the code is being written

- ... the options to complete the statement is shown
- **Initial error detection**
 - **Dynamic syntax checking [1]**
 - Highlight / underline / colour syntax errors as the code is entered
 - Type checking
 - Checking for used variables not declared / unused variables which are declared
- **Presentation**
 - Colour coding // pretty printing
 - **Expand and collapse code blocks [1]**
 - The user can hide code that they are currently not working on
 - **Auto-indentation [1]**
 - automatically indents // aids readability
- **Debugging [6]**
 - **Single-stepping [1~2]**
 - to allow program statements to be executed one at a line
 - ... so that the effect of each statement can be seen
 - **Breakpoints**
 - to pause / stop the program at a specific line / statement
 - **Variable / expression watch window [1]**
 - to monitor the value of variables / expressions as the program is run

6 Security, privacy and data integrity

6.1 Data Security

6.1.1 Explain the difference between the terms security, privacy and integrity of data

- **Security [2]**

- deal with protection of data / keeping data safe (*depending on the mark scheme writer*)
- protect data from illegal access / prevent data loss. / *personally recommend you to write data loss*

- **Privacy [2]**

- the need to restrict access to personal data
- To avoid it being seen by unauthorised people

- **Data integrity [2]**

- Ensure data is accurate / complete / consistent
- Ensure data is up to date
- Ensure data is not corrupted after, e.g. being transmitted.

- **Security versus privacy [2]**

- Security protects data against loss
- Privacy protects data against unauthorised access

6.1.2 Show appreciation of the need for both the security of data and the security of the computer system

- **Need of security for data [1]**

- To stop the data being lost / corrupted / amended
- To make sure it can be recovered
- To prevent unauthorised access
- **Need of security from unauthorised access**
 - **Data [1]**
 - Data needs protecting from someone amending / deleting or taking it
 - **Computer system [1]**
 - Computer system need protection to stop people for e.g. installing malware or damaging the system

6.1.3 Describe security measures designed to protect computer systems, ranging from the stand-alone PC to a network of computers

Including user accounts, passwords, authentication techniques such as digital signatures and biometrics, firewall, anti-virus software, anti-spyware, encryption

- **Passwords [1]**
 - Ensures only users with the correct password can access the resources // prevent unauthorised access
- **Authentication [2]**
 - process of determining whether people are what they claim to be
 - frequently done by log on passwords/biometrics
 - because passwords can be stolen, digital certification is used
 - helps to prevent unauthorised access to data
- **Digital signature**
 - **Explain [2]**
 - text put through a hashing algorithm to produce a digest

- Digest encrypted with sender's private key (to create the digital signature)
- the digital signature can only be decrypted with matching sender's public key
- **Explain [5]**
 - The sender hashes the document
 - ... to produce a digest
 - The sender encrypts the digest to create the digital signature
 - The message and the signature are sent to the receiver
 - The receiver decrypts the signature to reproduce the digest
 - The receiver uses the same hashing algorithm on the document received to produce a second digest
 - The receiver compares this digest with on from the digital signature
 - If both of the receiver's digest are the same the document is authentic
- **Firewall [2~3]**
 - monitors incoming and outgoing packets / traffic
 - checks against an allow list / deny list of IP addresses // checks against a set of rules for acceptable data / ports etc.
 - blocks transmissions that do not meet criteria / rules // allows through is satisfies the criteria /rules
- **Proxy server [2] (not in syllabus but occurred 2024)**
 - Prevents devices accessing the web server directly
 - Intercepts any requests
 - Forwards the request using its own IP address
 - Screens returning data before sending it to the user

- **Anti virus software [1]**

- scans the computer for viruses and checks against a stored database of viruses, that needs to be updated regularly and then deletes / quarantines them
- compares downloaded files to a database of known viruses and prevents the download continuing

- **Anti spyware [1]**

- scans the computer for spywares and checks against a stored database of spyware , that needs to be updated regularly and then deletes / quarantines them
- compares downloaded files to a database of known viruses and prevents the download continuing

- **Encryption [2~3]**

- Data is turned into cipher text // Data is encoded
- ... using an encryption key
- so that it cannot be understood if intercepted without the decryption key
- example e.g. VPN

- **Physical method [1]**

- e.g. the computer storing the data can't be accessed without the key to the room

6.1.4 Show understanding of the threats to computer and data security posed by networks and the internet

Including malware (virus, spyware), hackers, phishing, pharming

- **Malware**

- **Virus [2]**

- Malicious code / software /program

- that replicates / copies itself (*must write both two if defining the term*)
- can cause loss of data / corruption of data on the computer
- can cause computer to 'crash' / run slowly
- can fill up hard disk with data
- **Spyware [2]**
 - Software installed on the computer without the user's knowledge
 - secretly records keystrokes on computer
 - sends the data gathered to a third party
- **Comparing malwares**
 - **Virus & spywares**
 - **Similarity [2]**
 - Both are pieces of malicious software
 - Both are downloaded without user knowledge
 - Both can pretend to be / are embedded in other legitimate software when downloaded // both try to avoid firewall
 - Both run in the background
 - **Difference [1]**
 - Virus can damage computer data
 - Virus does not send data out of the computer
 - Virus replicates itself.
- **Hacker**
 - **Description [1]**
 - Unauthorised access to the computer with malicious intent
- **Phishing [2]**

- The email pretends to be from an official body
- ... persuading individuals to disclose private information // by example such as bank details
- ... or requesting authentication by redirecting to an unofficial/unauthorised website // inviting a user to click a link
- **Pharming**
 - **Describe [1]**
 - Malicious code/software installed on a computer which redirects user to a fake website to obtain personal data
 - Users are directed to a bogus website that looks legitimate to obtain personal information
 - **Information analysed from textbook**
 - malware installed to computer without user's knowledge
 - ... redirect user to a fake website
 - confidential information is encouraged to input
 - the information is sent to a 3rd party

6.1.5 Describe methods that can be used to restrict the risks posed by threats

- **Prevent accidental loss of data**
 - Frequent backup EITHER to secondary media / to 3rd party server
 - Disk-mirroring strategy / RAID
 - UPS(uninterruptable power supply) / backup generator
- **Data backup [2]**
 - A copy of data will have been made and stored elsewhere

- If the original is lost, the backup can be used to restore the data
- **Disk mirroring [2]**
 - The data is stored on two disks simultaneously
 - If the first disk drive fails, the data is accessed from the second disk

6.1.6 Describe security methods designed to protect the security of data [6]

Including encryption, access rights

- Installing a firewall and ensuring it is switched on.
- To stop unauthorised access / hackers gaining access
- Use authentication methods such as passwords and usernames.
- Passwords should be strong / biometrics
- Encrypt the data.
- So that if data is accessed it will be meaningless / only accessed by those with decryption key.
- Set up access rights
- To stop users reading / editing data they are not permitted to access
- Installing and running an up-to-date anti-malware program (anti-virus/anti-spyware etc.)
- To detect / remove / quarantine viruses / key-loggers etc.

- Make regular backups of the data
- To separate device or off site to enable recovery if necessary
- Employ measures for physical security
- example of a measure for physical security

6.2 Data Integrity

6.2.1 Describe how data validation and data verification help protect the integrity of data

- **Data validation**
 - ensure data is reasonable
 - examples: range checks, type checks, length checks...
- **Data verification**
 - checks if data input matches original / if transmitted data matches original
 - can use double entry or visual check / other methods such as parity checks
 - doesn't check whether data is reasonable

6.2.2 Describe and use methods of data validation

Including range check, format check, length check, presence check, existence check, limit check, check digit

- **Presence check [1]**
 - Checks the input is not blank
- **Range check [1]**
 - Check the number entered is inside a certain range
- **Format check [1]**

- Checks the input data is in a particular format
- **Length check [1]**
 - Check the number of characters of input data
- **Existence check [1]**
 - To ensure the data has been assigned
- **Check Digit [4]**
 - Multiply each digit with corresponding weight
 - Sum MOD 11 (or other number according to question)
 - Check digit is $(11 - (\text{Sum MOD } 11))$
 - 1 mark for correct final answer

6.2.3 Describe and use methods of data verification during data entry and data transfer

- During data entry including visual check, double entry
- During data transfer including parity check (byte and block), checksum
- **Visual check [1]**
 - Manually compare the data entered with the original
- **Double entry [1]**
 - Enter the data twice and the system compares them to see if they are the same
- **Parity check**
 - **Describe [4]**
 - even or odd parity which is defined before data sent
 - each byte has a parity bit
 - parity bit set to 0 or 1 to make parity for byte correct
 - after transmission, parity for byte re-checked
 - if different, an error is flagged

- parity block may identify the position of incorrect bit
- **Parity byte [1]**
 - An additional bit is added to make the number of 1s in the byte odd or even to match the parity. If a byte with an odd number of 1 bits is received when even parity is used, there is an error.
- **Parity block [1]**
 - Parity is calculated horizontally and vertically. A parity byte is created from the bits produced by the vertical parity check. This is sent with the data. The parity is re-checked when received and the position of an incorrect bit can be determined.
- **Why parity check may not work [2]**
 - Errors in an even number of bits (in the same row or column)
 - Prevents error being identified
 - Could appear to be correct
- **Parity Byte [2]**
 - A parity bit is worked out for each column
 - (After received) The computer checks the parity of each bit position in parity byte // the computer generates copy of the parity byte and compares
 - If incorrect parity then there is an error in the data received
 - The position of the incorrect bit can be determined
- **Checksum [4]**
 - calculation carried out on data to be sent
 - result is sent along with data
 - checksum is re-calculated at receiving end
 - compared, if same then no error
 - if different, data has been corrupted

- ... request is sent to re-send the data.
- **Hash total [2]** (no longer in syllabus)
 - Total of several fields of data
 - Including field not usually used in calculations
 - The result is transmitted with the data
 - Calculation repeated at receiving end
 - Results compared
 - If different an error has occurred

7 Ethics and Ownership

7.1 Ethics and Ownership

7.1.1 Show understanding of the need for and purpose of ethics as a computing professional

Understand the importance of joining a professional ethical body including BCS (British Computer Society), IEEE (Institute of Electrical and Electronic Engineers)

- **Describe what is meant by ethics [2]**
 - A system of moral principles
 - That guide behaviour / decision making
 - Based on philosophical / religious views
 - By example e.g. respectful and considerate behaviour
- **Principles of ACM/IEEE Software Engineering Code of Ethics [6]**
 - PUBLIC / Software engineers shall act consistently with the public interest
 - CLIENT AND EMPLOYER / Software engineers shall act in a manner that is in the best interests of their client and employer (consistent with the public interest)
 - PRODUCT / software engineers shall ensure that their products and related modifications meet the highest professional standards possible.
 - JUDGEMENT / Software engineers shall maintain integrity and independence in their professional judgment
 - MANAGEMENT / software engineering managers and leaders shall subscribe to and promote an ethical approach

to the management of software development and maintenance

- PROFESSION / Software engineers shall advance the integrity and reputation of the profession (consistent with public interest)
- COLLEAGUES / software engineers shall be fair to and supportive of their colleagues
- SELF / Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession.
- **Describe how to act in best interest of client [3]**
 - ... Keep the client's personal data private
 - ... Involve the client in the development // ... Communicate with the client
 - ... Provide the solution that the client asked for
 - ... Keep the project running on time // budget
 - ... Keep the client informed of any problems
- **Explain how to follow PRODUCT [2]**
 - Ensure product is of a high standard
 - Ensure product meets requirements
 - Ensure the product is delivered within time
 - Ensure the product is delivered within budget
 - Ensure product development is well-documented
 - Ensure product is tested thoroughly // free of bugs
- **Explain how to follow COLLEAGUE [2]**
 - Need to be fair// do not show any prejudices
 - To be supportive of colleagues
 - Work together as a team

- Listen to each other's ideas
- Assist colleagues in professional development
- Fully credit the work of other members of the team
- Make colleagues aware of expected standard ways of working

7.1.2 Show understanding of the need to act ethically and the impact of acting ethically or unethically for a given situation

- **Need for manager to act ethically [2]**
 - To make sure team members feel valued
 - To get the best work out of the team
 - To enable them to work well together
 - To enable them to create best product for the client
- **Need for programmer to join a professional ethical body [3]**
 - Has ethical guidelines to follow
 - so client/other staff know the standards being applied
 - so he does not have to decide what is ethical it's written down
 - Clients / staff know he is reputable
 - recognition of his skills / knowledge
 - there may be a test / requirement for entry
 - They provide help and support
 - for example if he needs legal advice
 - They run training courses
 - to keep his skills up-to-date
- **Ethical action before new staff starting work**
 - **Staff [1]**

- Read about the languages that will be used
- Visits the office prior to starting
- Speaks to manager about concerns
- **Manager [2]**
 - Prepare an induction
 - Invite new member in before starting
 - Introduce new member to the team
 - Give new member a mentor
- **Colleague [1]**
 - Prepare a greeting / introduction
 - Provide structured support
 - Invite new member to social event before starting to meet people

7.1.3 Show understanding of the need for copyright legislation

- **Copyright**
 - **Define [1]**
 - Formal or legal recognition of ownership of the program // Formal or legal restriction / permissions on use of the program // The intellectual property rights to the program
 - **Describe [2]**
 - The formal and legal rights to ownership // intellectual property rights
 - Protects against unauthorised reproduction of work
 - Provides for legal right of redress

7.1.4 Show understanding of the different types of software licensing and justify the use of a licence for a given situation

Licences to include Free Software Foundation, the Open Source Initiative, shareware and commercial software

- Open source

- Describe [1]

- Software can be freely copied, distributed or adapted
- *better define the term with the how the software can be used i.e. copied, distributed, adapted*

- Explain [2]

- The source code comes with the software
- The user can edit the source code
- Once edited, the software is re-distributed with the changes

- Justify choosing [2]

- allow users to customise the code
- Others may enhance / improve the program // allow collaboration
- allow errors to reported / identified / fixed by users
- allow additional features to be added to the code
- Can charge a fee for the App
- Does not have to set up ways to take funds

- Shareware

- Explain [2]

- User get a free trial of the software
- ... which may be limited in features
- No access to source code // Program cannot be edited

- Then they have to pay / sign-up after the expiry date //
- Then they have to pay / sign-up to get full functionality //
- Then they have to pay / sign-up to stop unwanted pop-ups, etc.

- **Benefits [2]**

- The user can test/trial the program before purchasing to see whether it meets needs
- There is no initial cost
- Users who try the software may spread the word so the developer gets more custom
- The developer can get feedback to improve the program

- **Justify choosing (developer) [2~3]**

- Enables the program to be copyrighted
- The user cannot legally edit the program so the developer retains control over product
- User can try the program for free and then pay for the full game which allows the programmer to gain an income
- so more people can experience it and therefore be more likely to buy it
- Users can try the program and offer constructive feedback

- *Note: Shareware can be redistributed*

- **Commercial software**

- **explain [2]**

- The software is purchased // anyone can purchase if agree to the terms
- With a licence which restricts the number of users / possible time period of use

- The program code for the software cannot be edited // **source code not provided**
- **Benefit [2]**
 - The owner/developer can earn a fee
 - Comprehensive support is usually offered so problems can be easily solved
 - The software is usually well tested so few bugs
- **Justify choosing (company) [4]**
 - Support / training is readily available so help can be accessed if needed
 - More robust software / fewer bugs as it has been tested more thoroughly
 - Forums / user groups will exist for popular software
 - Software upgrade path likely to be available (at minimal cost)
 - Manufacturer develops patches that can be automatically downloaded
 - Compatibility is inbuilt for other commercial software.
- **Software License [1]**
 - A (legal) agreement that defines the rights and terms of usage for the software
- **Free Software Foundation [2]**
 - is free of restrictions
 - not necessarily free of charge
 - a user can edit source code
 - user can share software with others
 - *note: it is not necessarily free of charge instead of free of charge*
- **Freeware [3] (no longer in syllabus)**

- No fee charged
- Software is copyrighted
- Modification, re-distribution or reverse engineering without permission is prohibited

7.1.5 Show understanding of Artificial Intelligence (AI)

- Understand the impact of AI including social, economic and environmental issues
- Understand the applications of AI
- **Social impacts of AI**
 - **Facial recognition in airport [2]**
 - incorrect recognition of faces leads to mistakes such as
 - ... access to facilities / system may be denied
 - privacy issues / people do not like data being stored
 - individuals will feel safer
 - ... there might be a reduction in crime
 - faster boarding
 - catching criminals
- **Economic impacts of AI**
 - **Car diagnosis program in garage [2]**
 - reduce cost to the garage
 - ... because less time taken for diagnosis
 - increase profits for the garage
 - ... as technicians spend more time repairing, so completing more jobs in a day
 - decrease costs passed to customer
 - ... so garage may gain customers

- profit margins can be reduced
- ... because program may be expensive to buy / maintain / update

- **Applications of AI [3]**

- Police identifying wanted people
- Uses image recognition
- ... to identify features/characteristics/items in an image

- Natural language interfaces
- Use speech recognition to identify words that are spoken
- ... and adapts to learn regional accents

- Self-driving cars
- Detects its position on the road and within the traffic
- Follows a route // Collision avoidance // Self-parking etc.

- Spoken Interfaces
- Use natural language processing
- ... to take a sentence and work out its meaning

- Game playing
- Models characters in a computer game
- ... to allow computer characters to react according to the player's movements

- **How AI Play game [3]**

- The rules / past moves / decision making algorithms of the game will be stored
- The AI program is trained, by playing many times

- AI will look (ahead) at possible moves
- and/or analyse the pattern of past choices
- and choose the move most likely to be successful
- Computer could learn how to improve // learn from previous mistakes
- by storing the positive/negative result of choices
- and changing its future choices
- **Facial recognition [4]**
 - Captures an image of the face
 - Uses image recognition
 - Trained to identify the features of a face in an image
 - ... using large number of images
 - Analyse images for facial features
 - Uses the probability to match
- **CCTV system with AI tracking person in house [4]**
 - Uses image recognition
 - Monitors every image taken to identify matching images / shapes / features to a 'person'...
 - ... starts recording to secondary storage/permanently when a person is identified
 - System identifies direction of movement of person and uses this to decide where/how to move the camera/record
 - System identifies other cameras to start recording based on direction of movement
- **AI control camera to focus on people's face [3]**
 - Scans the scene in real time
 - Identifies if there are faces in the image
 - Uses facial recognition
 - ... uses image recognition

- ... takes each frame individually
- ... analyses the pixels
- ... stores pattern for a face
- ... looks for patterns that match/come close to the pattern for a face
- Camera focuses on the pattern identified
- **AI recognise car registration number [4]**
 - It uses image recognition
 - The pixels of each image from the camera are stored
 - ... and matched to the expected shape/size/colour of registration number
 - It uses optical character recognition
 - The pixels within the registration number area identified are analysed
 - ... and compared to expected characters/letters/numbers
- **AI convert speech to command [3]**
 - Uses speech recognition
 - ... which identifies key phrases / words spoken
 - ... and matches these to a database
 - ... and generates the most likely sentence / command / word
- **AI ordering food [3]**
 - Voice/speech recognition is used
 - ... to identify if someone speaking
 - The sound is recorded and analysed
 - The audio recordings are compared to a database of words/ sound waves
 - ... to identify the word that has the highest probability of being said
 - Natural language recognition is used

- Words are combined and compared to known sentences
- ... programmed action for matching sentence are performed

8 Databases

8.1 Database Concepts

8.1.1 Show understanding of the limitations of using a file-based approach for the storage and retrieval of data

- Explain [3]

- Data redundancy
 - because the same data is stored many times
- Program-data dependency
 - because any changes to the structure of the data means the programs that access that data have to be re-written
- Data inconsistency // bad data integrity
 - because duplicated data might be stored differently // because when data is uploaded in one place, it is not updated everywhere
- Not easy to perform complex searches /queries
 - because a new program has to be written each time
- Lack of privacy
 - as user views cannot easily be implemented

8.1.2 Describe the features of a relational database that address the limitations of a file-based approach

- Describe [3]

- By storing data in (separate) linked tables data redundancy is reduced / data duplication is controlled...
- Data integrity issues are reduced as data only needs to be updated once / is only stored once

- Unwanted or accidental deletion of linked data is prevented as the DBMS will flag an error
- Program - data dependence is overcome
- Changes made to the structure of the data have little effect on existing programs
- Complex queries can be more easily made as the DBMS will have a query language / QBE form
- Unproductive maintenance is eliminated as changes only need to be made once (rather than changing multiple programs)
- Fields can be added or removed without any effect on existing programs (that do not use these fields)
- Security / privacy of the data is improved as each application only has access to the fields it needs
- Better control over data integrity as the DBMS (uses its Data Dictionary) to perform validation checks in data entered.
- **How data redundancy is addressed [3]**
 - Each record / price of data is stored once and is referenced by a (primary) key
 - Because data is stored in individual tables
 - ... and the tables are linked by relationships
 - By the proper use of Primary and Foreign keys
 - By enforcing referential integrity

8.1.3 Show understanding of and use the terminology associated with a relational database model

Including entity, table, record, field, tuple, attribute, primary key, candidate key, secondary key, foreign key, relationship (one-to-many, one-to-one, many-to-many), referential integrity, indexing

- **Entity [1]**

- Any object, person or thing about which it is possible to store data // *A real-life object that is represented as a table*
- *(depending on the preference of the examiner)*
- **Table [1]**
 - Dataset organised in rows and columns; the columns form the structure and the rows form the content.
- **Field [1]**
 - A column / attribute in a table
- **Tuple [1]**
 - a single row in a table
- **Attribute [1]**
 - An item of data about an entity
- **Primary key [1]**
 - Attribute or combination of attributes that is used to uniquely identify a record
- **Candidate key [1]**
 - An attribute / field (or set of attributes / fields) that could be a primary key
- **Secondary key**
 - **Describe [1]**
 - An additional/alternative key used as well as the primary key to locate specific data // a candidate key that has not been chosen as a primary key.
 - **Reason to implement a secondary key [2]**
 - Allows for a faster searching using the attribute
 - Since this attribute will be frequently searched
 - This attribute has been indexed
- **Foreign key [1]**

- Attributes in a table that link to a primary key in another table to form a relationship
- **how to implement relationship between two tables [2]**
 - identify the primary key in the first table
 - link this primary key to second table
 - so that it becomes the foreign key in the second table
- **Referential integrity**
 - **Explain [3]**
 - Ensure related data in tables are consistent
 - Making sure tables do not try to reference data which does not exist
 -
 - Referential integrity ensures that every foreign key has a corresponding primary key
 - If one table has a foreign key (the 'foreign table')
 - ... then it is not possible to add a record to that table
 - ... unless there is a corresponding record in the linked table with a corresponding primary key (the 'primary' table)
 -
 - Cascading delete
 - If a record is deleted in the 'primary' table ...
 - all corresponding linked records in 'foreign' tables must also be deleted
 -
 - Cascading update
 - If a record in the 'primary' table is modified...
 - ... all linked records in foreign tables will also be modified

- Why important [3]

- makes sure data is consistent
- makes sure data is up-to-date
- ensures that every foreign key has a corresponding primary key
- prevents records from being added/deleted/modified incorrectly
- makes sure if data is changed in place the change is reflected in all related records
- makes sure any queries return accurate and complete results

- Indexing (*analysed from DeepSeek*)

- DBMS create sorted reference structure (e.g. B-tree or hash table)
- ... that stores the key values along with pointers to their actual locations in the table
- When a query searches for a value, the DBMS checks the index
- once found, the index provides a direct pointer to the corresponding row(s) in the table, speeding up the retrieval
- when a new row is inserted, the DBMS must update the index

8.1.4 Use an entity-relationship (E-R) diagram to document a database design

try

- 9618_w23_qp_11 Q3 (a)

8.1.5 Show understanding of the normalisation process [3]

- First Normal Form (1NF),
- Second Normal Form (2NF) and
- Third Normal Form (3NF)
- 1NF
 - No repeated groups of attributes
 - All attributes should be atomic
 - No duplicate rows
- 2NF
 - No partial dependency
- 3NF
 - No non-key dependency
 - No transitive dependency
- *note: try 9618_w23_qp_12 Q2(c)*

8.1.6 Explain why a given set of database tables are, or are not, in 3NF

- **Not in 3NF [2]**
 - there is non-key dependency
 - (identify the dependency)

8.1.7 Produce a normalised database design for a description of a database, a given set of data, or a given set of tables

8.2 Database Management System (DBMS)

8.2.1 Show understanding of the features provided by a Database Management System (DBMS) that address the issues of a file based approach

Including:

- data management, including maintaining a data dictionary
- data modelling
- logical schema
- data integrity
- data security, including backup procedures and the use of access rights to individuals / groups of users

- Data dictionary

- Describe [1]

- Data about the data in the database // data about the structure of the database // metadata for a database

- Describe [2]

- A file/table containing all the data about the detail of the database design
- For example, fields, data types, validation, keys

- Example content [3]

- table name
- field name // attribute

- data type
- type of validation
- Primary key
- Foreign key
- relationships
- **Logical schema**
 - **Describe [1]**
 - A model of a database that is not specific to one DBMS
 - **Describe [2]**
 - the overview of a database structure
 - models the problem / situation
 - ... by using methods such as ER diagram
 - independent of any particular DBMS
 - 2024 answer:*
 - Conceptual design
 - Platform/database independent overview of the database
 - used to design the physical structure
 - Appropriate example e.g. Design of entities / E-R diagram / views
 - **Levels of logical schema [4]**
 - External
 - The individual's view of the database
 - Conceptual
 - Describes the data as seen by the applications making use of the DBMS
 - Describes the 'views' which users of the database might have
 - Physical / Internal

- Describes how the data will be stored on the physical media
 - Logical
 - describes how the relationships will be implemented in the logical structure of the database
- **Data integrity**
 - **Define [1]**
 - Methods of making sure the data is consistent
 - **Example of implementation [2]**
 - Enforcing referential integrity
 - If data in one table is deleted/edited all tables are updated // cascading update/delete
 - Validation/verification rules
- **Data security [4]**
 - Issue usernames and passwords...
 - stops unauthorised access to the data
 - further expansion e.g. strong passwords / passwords changed regularly etc.
 - Access rights / privileges
 - so that only relevant staff / certain usernames can read/edit certain parts of the data
 - can be read only, or full access / read, write and delete
 - Create (regular / scheduled) backups
 - in case of loss/damage to the live data a copy is available
 - Encryption of data
 - if there is unauthorised access to the data it cannot be understood // need a decryption key
 - Definition of different view...

- composed of one or more tables
- control the scope of the data accessible to authorised users
- Usage monitoring / logging of activity...
 - creation of an audit / activity log
 - records the use of the data in the data base / records operations performed by all users / all access to the data

8.2.2 Show understanding of how software tools found within a DBMS are used in practice

Including the use and purpose of:

- developer interface
- query processor

- **Developer interface**

- **Description [1]**

- A software tool that allows the user to create items such as tables, forms and reports // enable developer to create user-friendly forms and reports

- **Explain how to make use of // purpose of [3]**

- To create / modify / delete tables
- To create a form for data input
- To add a menu to enable users to choose different actions
- To add tools to a form
- for example, drop-down boxes / buttons etc.
- To modify relationships

- **Query processor**

- **Describe [1]**

- Software that allows the user to enter criteria, then finds and returns the appropriate result // software that processes and executes queries written in SQL
- **Describe [2]**
 - Set up search criteria
 - To find / retrieve / return the data that matches the criteria
 - Organise the results to be displayed to the user
- **How DBMS supports data integrity [2]**
 - Validation
 - Enforcing referential integrity
 - Cascade update/delete
 - Ensuring the database is normalised

8.3 Data Definition Language (DDL) and Data Manipulation Language (DML)

8.3.1 Show understanding that the DBMS carries out all creation / modification of the database structure using its Data Definition Language (DDL)

8.3.2 Show understanding that the DBMS carries out all queries and maintenance of data using its DML

8.3.3 Show understanding that the industry standard for both DDL and DML is Structured Query Language (SQL)

[Understand a given SQL statement](#)

8.3.4 Understand given SQL (DDL) statements and be able to write simple SQL (DDL) statements using a sub-set of statements

Create a database (CREATE DATABASE)

Create a table definition (CREATE TABLE), including the creation of attributes with appropriate data types:

- CHARACTER
- VARCHAR(n)
- BOOLEAN
- INTEGER
- REAL
- DATE
- TIME

change a table definition (ALTER TABLE)

add a primary key to a table (PRIMARY KEY (field))

add a foreign key to a table (FOREIGN KEY (field)

REFERENCES Table (Field))

- Create database [1]

- CREATE DATABASE <Name>;

- Create table [6]

- CREATE TABLE <table> (...);
- one mark per data type
- one mark for primary key (*if have*)
 - and NOT NULL for primary key if varchar
- one mark for foreign key (*if have*)
- e.g.

```
CREATE TABLE STUDENT_TEST (  
    StudentID INTEGER,
```

```

TestID VARCHAR,
Mark INTEGER,
PRIMARY KEY(StudentID, TestID),
FOREIGN KEY(TestID) REFERENCES TEST(TestID)
FOREIGN KEY(StudentID) REFERENCES STUDENT(StudentID)
);

```

- Alter table [2]

- ALTER TABLE USER
- ADD Country VARCHAR(25)

- ALTER TABLE EVENT
- ADD FOREIGN KEY(PlayerID)
- REFERENCES PLAYER(PlayerID)
-
- ALTER TABLE <name>
- DROP COLUMN <attribute>
-
- ALTER TABLE
- ALTER COLUMN <attribute> TYPE <type>

8.3.5 Write an SQL script to query or modify data (DML) which are stored in (at most two) database tables

- Queries including SELECT... FROM, WHERE, ORDER BY, GROUP BY, INNER JOIN, SUM, COUNT, AVG
- Data maintenance including. INSERT INTO, DELETE FROM, UPDATE

- Queries

- SELECT <fields> (COUNT(*), AVG(field), SUM(<field>)
- FROM <table> / <table1> INNER JOIN <table2>
- (ON <table1>.primary_key = <table2>.foreign_key)
- WHERE <condition>
- GROUP BY <field> ASC / DESC

- WHERE <field> LIKE 'HW%' looks for those begin with HW

- Manipulation

```
INSERT INTO <table> (<field1>, <field2>, ...)  
VALUES (<value1>, <value2>, ...)
```

```
DELETE FROM <table>  
WHERE <condition>
```

```
UPDATE <table>  
SET <field> = <value>  
WHERE <condition>
```

- Dealing with dates

- Mark schemes are in general very ambiguous about the extraction of date in the SQL. Here, I'll display several formats that occurred in past papers.

- Extraction of month

```
WHERE ReleaseDate >= '01/01/2022' AND  
ReleaseDate <= '01/31/2022';
```

```
WHERE ReleaseDate >= #01/01/2022# AND  
ReleaseDate <= #01/31/2022#;
```

```
WHERE ReleaseDate BETWEEN '01/01/2022' AND  
'31/01/2022';
```

```
WHERE ReleaseDate BETWEEN #01/01/2022# AND  
#31/01/2022#;
```

- Extraction of date

- WHERE BirthDate = '01/01/2022'

9 Algorithm Design and Problem-Solving

Refer to Pseudocode Guide <https://www.cambridgeinternational.org/Images/697401-2026-pseudocode-guide-for-teachers.pdf>

9.1 Computational Thinking Skills

9.1.1 Show an understanding of abstraction

- Need for and benefits of using abstraction
 - Describe the purpose of abstraction
 - Produce an abstract model of a system by only including essential details
- **Abstraction**
 - **Describe [1]**
 - Filter out information that is not essential when designing the algorithm
 - **Benefit [1]**
 - The solution is simplified so easier / quicker to design / implement
 - The system is tailored to the need of the system (*depend on context*)

9.1.2 Describe and use decomposition

Break down problems into sub-problems leading to the concept of a program module (procedure / function)

- **Describe decomposition [2]**
 - Breaking down a problem into sub problems / smaller parts

- In order to explain / understand // easier to solve problem
- Leading to the concept of program modules // assigning problem parts to teams

9.2 Algorithms

9.2.1 Show understanding that an algorithm is a solution to a problem expressed as a sequence of defined steps

- **Explain the term algorithm [2]**

- A sequence of steps / stages / instructions
- to implement a task // solution to a program

9.2.2 Use suitable identifier names for the representation of data used by a problem and represent these using an identifier table

You can do this.

9.2.3 Write pseudocode that contains input, process and output

- **Input [2]**

- Enter data into the system
- `INPUT MyVar // READFILE MyFile, MyString`

- **Process [2]**

- Manipulate / change data in some way // perform calculation / find a result
- `MyChar ← 'X' // MyNum ← MyNum + 1`

- **Output [2]**

- Send data out from the system // display / print / transmit / show data
- `OUTPUT "Hello World" // WRITEFILE MyFile, MyString`

9.2.4 Write pseudocode using the three basic constructs of sequence, selection and iteration (repetition)

- Selection:

```
CASE OF MyVar
    <VALUE> : <CODE>
    <VALUE> : <CODE>
    OTHERWISE : <CODE>
ENDCASE
```

```
IF <CONDITION> THEN
    <CODE>
ELSE
    <CODE>
ENDIF
```

9.2.5 Document a simple algorithm using a structured English description, a flowchart or pseudocode

- **Describe the purpose of an algorithm [2] (no longer in syllabus)**
 - Linear search:
 - To search for a given value in the array
 - and output the position in the array if the search value is found
 - if search value is not found output message
 - Cambridge

- **Design an algorithm**

- **Linear search [4]**

- A loop/ repetition/ iteration to check every element
 - Compare each element with the value being searched
 - Exit loop / stop search when value found or end of array reached
 - If value is found then output the index position, otherwise output 'Not found', etc.

- **Count number of lines in a file [3]**

- Open the file
 - Set a count to zero
 - Loop until end of file // no more lines to read
 - Increment the count each time a line is read in a loop

- **Read and store lines from a file [5]**

- Open file
 - Initialise an index variable
 - Repeat until the end of file is reached
 - Read a line from the file
 - Store the line / variable in the array at the index
 - Increment the index in a loop

9.2.6 Write pseudocode from:

9.2.6.1 a structured English description

9.2.6.2 a flowchart

9.2.7 Draw a flowchart from:

9.2.7.1 a structured English description

9.2.7.2 pseudocode

9.2.8 Describe and use the process of stepwise refinement to express an algorithm to a level of detail from which the task may be programmed

- **Stepwise refinement**

- **Describe [2]**

- to increase the level of detail of an algorithm / design... // breaking down a problem / module / task into smaller parts
 - ... from which the task may be programmed.

- **Advantage [2]**

- Makes the problem / task / algorithm easier to understand // reduce problem complexity
 - Smaller modules easier to develop / test / debug
 - Programmers can work on different modules // different expertise

9.2.9 Use logic statements to define parts of an algorithm solution

10 Data Types and structures

10.1 Data Types and Records

10.1.1 Select and use appropriate data types for a problem solution

including integer, real, char, string, Boolean, date (pseudocode will use the following data types:

INTEGER, REAL, CHAR, STRING, BOOLEAN, DATE, ARRAY, FILE)

- When to use boolean [1]

- To terminate a (conditional) loop when a value has been found
- When the variable can take only one of two possible values
- (Accept by example): when a variable is recording when an action has been done e.g. Yes or No // light is on

10.1.2 Show understanding of the purpose of a record structure to hold a set of data of different data types under one identifier

- Write pseudocode to define a record structure
- Write pseudocode to read data from a record structure and save data to a record structure

- Record structure [3]

- one mark for TYPE and ENDTYPE
- one mark per data type
- e.g.

```
TYPE STUDENT
    DECLARE StudentID : STRING
    DECLARE Age : INTEGER
ENDTYPE
```

- Benefit of record [2]

- can store mixed data types / multiple data types under single identifier
- Tighter association between field // simpler code as fields may be referenced together // values can't get out of step as with two arrays
- Program is easier to design / write / debug / test / maintain

10.2 Arrays

10.2.1 Use the technical terms associated with arrays

Including index, upper and lower bound

The following pseudocode includes references to a 1D array.

```

DECLARE StudentGrade : ARRAY[1:5] OF CHAR
DECLARE n : INTEGER
DECLARE x : CHAR

n ← 3
x ← StudentGrade[n]
```

(i) Use the correct technical terms to explain the meaning of [1:5] in this pseudocode.

.....

.....

.....

.....[2]

(ii) Use the correct technical term to complete the following statement.

Integer n is used as the to StudentGrade. [1]

Answer	Marks
- The array is 1D 1 is the lower bound 5 is the upper bound - size of array / number of elements = 5	Max2
subscript / index	1

- **Why sth e.g. -1 is chosen as initial array index [1]**

- Invalid as array index

10.2.2 Select a suitable data structure (1D or 2D array) to use for a given task

- **Benefit of array [2~3]**

- simplifies the algorithm // easier to write/ understand / test / debug
- It is possible to iterate through the variables // can use a loop // allow the storage of many values using a single identifier
- Use of index to access an element in the array
- Less identifiers needed
- fewer lines of code needed

- **Benefit and drawbacks of separate lines[2]**

- Benefits
 - algorithm to store/extract a record e.g. is easier
- Drawback
 - more file accesses data are needed (to read / write a complete record)

10.2.3 Write pseudocode for 1D and 2D arrays

You should be able to do this.

10.2.4 Write pseudocode to process array data

- Sort using a bubble sort
- Search using a linear search

You should be able to do this.

10.3 Files

10.3.1 Show understanding of why files are needed [1~2]

- Information is saved after the program ends // after the computer switched off
- so e.g. the game can continue from the saved point
- The amount of values stored in a text file is not fixed
- **Use of meaningful format of filename[2]**
 - *Not in the syllabus but appear in w25_23*
 - include suffix based on <based on the question>
 - concatenate with a root filename such as <based on the question>

10.3.2 Write pseudocode to handle text files that consist of one or more lines

- **Use of a special character [2]**
 - To provide a separator between data items
 - The special character does not appear in the data
 - Algorithm to extract / locate individual items from a line of text is simplified
- **Example Code [5]**

```
OPENFILE "PRODUCTS.txt" FOR READ
i <- 1
WHILE NOT EOF("PRODUCTS.txt") :
    READFILE "PRODUCTS.txt", PCode[i]
    READFILE "PRODUCTS.txt", PDescription[i]
    ...
    i <- i+1
ENDWHILE
```

10.4 Introduction to Abstract Data Types (ADT)

10.4.1 Show understanding that an ADT is a collection of data and a set of operations on those data

10.4.2 Show understanding that a stack, queue and linked list are examples of ADTs

Describe the key features of a stack, queue and linked list and justify their use for a given situation

- Queue

- Key features [3]

- Each queue element contains one data item
- A pointer to the front / start of the queue
- A pointer to the back / end of the queue
- Data is added at back / end and removed from front / start // works on a FIFO basis
- may be circular
- manipulate by using enqueue()/dequeue()

- Linked list

- Key features [3]

- Each node contains data and a pointer to the next node
- A pointer to the start of the list
- Last node in the list has a null pointer
- Data may be added / removed by manipulating pointers (not moving data)
- Nodes are traversed in a specific sequence

- Unused nodes are stored in a free list // a free list pointer to the free list
- **Advantage (to 1D array) [2]**
 - Pointers determine the ordering of data // only the pointers need to be changed when data is changed
 - Easier to add / delete data (to maintain correct sequence) in a linked list
- **Disadvantage (to 1D array) [2]**
 - Need to store pointers as well as data
 - More complex (to setup / implement)

10.4.3 Use a stack, queue and linked list to store data

Candidates will not be required to write pseudocode for these structures, but they should be able to add, edit and delete data from these structures.

10.4.4 Describe how a queue, stack and linked list can be implemented using arrays

- **Queue**
 - **Setup [5]**
 - Declare a 1D array of required size
 - ... of required data type
 - Declare integer variable for `FrontOfQueuePointer`
 - Declare integer variable for `EndOfQueuePointer`
 - Initialise `FoQ` and `EOQ` to represent an empty queue
 - Declare integer variable `NumberInQueue`
 - Declare integer variable for `SizeOfQueue` to count /limit the max number of items allowed // reference to mechanism for defining 'wrap' of circular queue

- Initialise `SizeOfQueue` // Initialise `NumberInQueue`
- **Add element [2]**
 - `EoQ` incremented
 - Data stored in location pointed to by `EoQ`.
- **Add element (further) [3]**
 - If incremented `EoQ=FoQ` the error condition: queue is full
 - Increment `EoQ`
 - Manage wrap-around
- **Remove element [1]**
 - *(when queue is not empty)*
 - `FoQ` pointer incremented
- **Condition when `FoQ` and `EoQ` have same value [1]**
 - Only one data item in the queue
- **Stack**
 - **Implementation [5]**
 - Declare a (1D) array of data type `STRING`
 - The number of elements in that array corresponds to the size of the required stack
 - Declare an integer / variable for `StackPointer`
 - Declare an integer / variable for the size of the stack // for the max value of `StackPointer`
 - Use the `StackPointer` as an index to the array
 - Pointers and variables initialised to indicate empty stack
 - Store each item on the stack as one array element / Each stack item maps to one array element
 - Attempt to describe Push and Pop operations
 - Push and Pop routines need to check for full or empty conditions

- e.g. push: add element to the top of the stack and pop: removes the top element from the stack
- **Why no need to initialise array [2]**
 - Unused values cannot be popped / taken off the stack // initialised values would never be used / unused elements cannot be accessed
 - ... until a value has first been pushed / written // overwrites previous value
- **Linked list**
 - **Setup [4]**
 - Declare two 1D arrays
 - One for data, one for pointer
 - Element from same index represent one node
 - Declare an integer / variable for StartPointer
 - Define appropriate value for null pointer
 - Declare an integer / variable for FreeList pointer
 - Alternative for the first 3 points:*
 - Define a record type with fields for data and pointer
 - Declare one 1D array
 - ... of the defined record type
 - **Iterate through [4]**
 - Assign `HeadPointer` to `ThisPointer` / `Current Pointer` // Identify first node using headpointer.
 - Loop the following until `ThisPointer` value is 0 / null pointer reached

- ... output data from array element at index
`ThisPointer` // Output data of the current node / array element
- ... assign pointer field from current array element / index / to `ThisPointer` / Current Pointer // Assign pointer field from current node to `ThisPointer` / Current Pointer

11 Programming

11.1 Programming Basics

11.1.1 Implement and write pseudocode from a given design presented as either a program flowchart or structured English

You'll have to practice for this one

11.1.2 Write pseudocode statements for:

- Constants

- `CONSTANT MinDiscount = 0.1`

- Advantage of using constant [3]

- values are entered once only
- avoid / minimise errors / changing the value accidentally
- when required, the constant value is changed once // Easier to maintain / update the program
- Makes the program / code easier to understand

11.1.2.2 the declaration of variables

`DECLARE <identifier> : <Data type>`

11.1.2.3 the assignment of values to variables

`<identifier> <- <value>`

11.1.2.4 expressions involving any of the arithmetic or logical operators input from the keyboard and output to the console

`INPUT <identifier>`

`OUTPUT <expression>`

11.1.3 Use built-in functions and library routines

Any functions not given in the pseudocode guide will be provided

String manipulation functions will always be given

- **Advantage of Built in functions // library routines [3]**
 - Saves development time / no need to write it / can't write it...
 - Pre-compiled and tested / Increased reliability / reduces chance of error
 - Available to all programs
 - No maintenance of function//routine is needed
- **Compare built-in function and a user-defined function**
 - **Difference [2]**
 - Built-in functions are made available by the programming language / already in the system
 - Built-in functions are ready made and tested
 - User-defined functions can be modified // built-in can't be modified
 - User-defined functions can be designed to meet the user's requirements
 - User-defined functions can only be used in that program / module

- Common [2]

- They have identifier name
- They return value
- They have none, one or more arguments
- Both perform specific task
- Both represent re-usable code
- Both are 'called'

11.2 Constructs

11.2.1 Use pseudocode to write:

11.2.1.1 an 'IF' statement including the 'ELSE' clause and nested IF statements

```
IF <condition> THEN
    <body>
ELSE
    <body>
ENDIF
```

11.2.1.2 a 'CASE' structure

```
CASE OF <identifier>:
    <value1> : <statement1>
               <statement2>
    <value2> TO <value3> : <statement3>
    OTHERWISE : <statement4>
ENDCASE
```

- *note: CASE clauses are checked in sequence, so if one condition is satisfied, the following will not be checked.*

11.2.1.3 a 'count-controlled' loop:

```
FOR    <identifier> <- <value1> TO <value2>
```

```
    <statement>
    (BREAK)
NEXT <identifier>
```

```
FOR <identifier> <- <value1> TO <value2> STEP
<increment>
    <statement>
    (BREAK)
NEXT <identifier>
```

- note: BREAK syntax is not listed in pseudocode guide, but it appears in both examiner report and w25_23 mark scheme

11.2.1.4 a 'post-condition' loop

- **E.g. write post-condition loop to output every odd number between 100 and 200 [4]**
 - Counter declaration and initialisation to sensible value before the loop.
 - REPEAT ... UNTIL <condition>
 - Correct selection of number to be output
 - Correct range of numbers output

11.2.1.5 a 'pre-condition' loop

```
WHILE <condition>
    <statement>
ENDWHILE
```

11.2.2 Justify why one loop structure may be better suited to solve a problem than the others

- **conditional loop [1]**
 - number of iteration is not known
- **count-controlled loop [1]**
 - number of iteration is known/given

11.3 Structured Programming

11.3.1 Define and use a procedure

```
PROCEDURE <identifier> ((BYREF) <param>:<data type>)  
    <statements>  
ENDPROCEDURE
```

11.3.2 Explain where in the construction of an algorithm it would be appropriate to use a procedure

- **Subroutine**
 - **Advantage in algorithm construction [1]**
 - To make a more manageable / understandable solution
 - To support modular design
 - **Advantage in program [3]**
 - Allows the subroutine to be called from many / multiple places
 - Subroutine may be (independently) tested and debugged
 - If the task changes the change needs to be made only once
 - Reduce unnecessary duplication / program lines
 - Allows teams to work on different parts of the solution
 - **Reason to use [2]**
 - When a selection of the code would be repeated
 - When a piece of code is needed to perform a specific task
 - To support modular programming / step-wise refinement
 - Easier to debug

11.3.3 Use parameters

A procedure may have none, one or more parameters

A parameter can be passed by reference or by value

- Usage of parameter [3]

- to pass values to/from the subroutine
- to produce re-usable code
- to avoid global variables
- to allow recursion

- Describe by reference [1]

- The address of / pointer to the parameter is passed to the subroutine // if the parameter value is change in the subroutine this changes the original value
- any change made to the parameter within module is reflected in the value in the calling module

- By value [1]

- any change made to the parameter value within module is NOT reflected in the value in the calling module
- A new instance of variable is created each time the procedure is called / executed
- So the previous contents are lost

11.3.4 Define and use a function

```
FUNCTION <name> (<param>:<type>) RETURNS <type>
    <statements>
    RETURN <value/identifier>
ENDFUNCTION
```

11.3.5 Explain where in the construction of an algorithm it is appropriate to use a function

A function is used in an expression, e.g. the return value replaces the call

- When a selection of the code would be repeated
- When a piece of code is needed to perform a specific task
- **How to change a section of code into a function [3]**
 - Create a module / function / procedure / subroutine using the (old) code / algorithm
 - Replace the old code / algorithm wherever it appears with a call to the module / function / procedure / subroutine
 - Pass the data as parameters // use of global variables
 - Return the result (of calculation) // Assign result to global variables / BYREF parameters
 - Create local variables as needed (to perform the calculation)

11.3.6 Use the terminology associated with procedures and functions

including procedure / function header, procedure / function interface, parameter, argument, return value

- **Interface [2]**
 - Provides a mechanism to allow calling program to pass data
 - Defines the parameters of procedure/function
 - ... giving their data type and order

11.3.7 Write efficient pseudocode

- **Why ordered data may improve efficiency [3]**
 - The algorithm / search / iteration can stop /only iterates
 - if the current value
 - is greater than the value being searched for

12 Software Development

12.1 Program Development Lifecycle

12.1.1 Show understanding of the purpose of a development life cycle

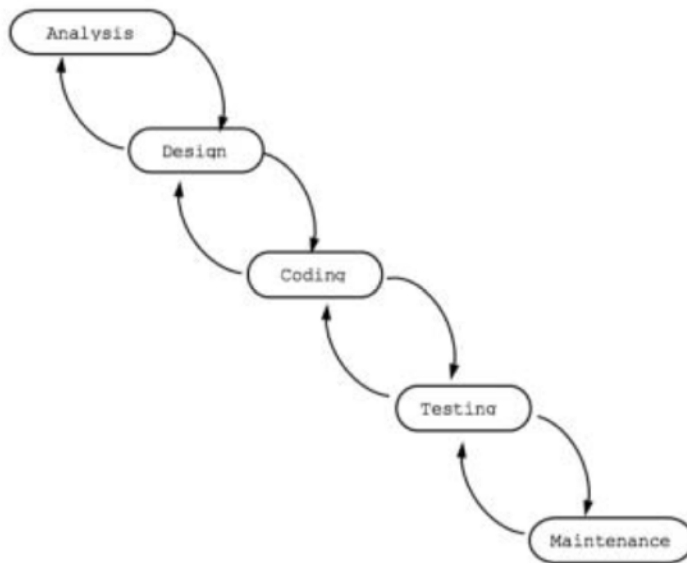
- **Benefit of program development lifecycle [1~2]**
 - Easier to manage / plan / cost // Clear deliverables produced at (end of) each stage
 - Clear deliverables // prototypes produced at (end of) each stage

12.1.2 Show understanding of the need for different development life cycles depending on the program being developed

Including: waterfall, iterative, rapid application development (RAD)

12.1.3 Describe the principles, benefits and drawbacks of each type of life cycle

- **Waterfall model**
 - **Principle [3]**



-

- Downward arrows : result from on stage is input / passed to the next stage
- Upward arrows : more work is required from previous stage to complete the current stage

- **Disadvantage [2]**

- No working software until late in the life cycle so slower to market than competitors
- More difficult to cope with changes to the requirements
- Low involvement/feedback of the client

- **Rapid application development (RAD)**

- **Principle [1]**

- Modules are developed in parallel / as prototypes
- Minimal / no detailed planning is carried out // Allows for changes to requirements
- Flexible development process
- Small incremental releases are made, each adding functionality
- Used for time critical development
- Client involved (all stages) of development

- **Advantage [2]**

- Quicker development possible / Multiple areas can be worked on at same time
- Prototype produced (at early stage in process)
- Easier to change requirements / quicker delivery of usable modules
- Early review possible / closer cooperation between client and developers

- **Disadvantage [1]**

- Difficult to estimate time / cost to complete project
- Documentation often omitted
- Lack of client availability throughout life cycle // too easy for client to keep changing their mind

12.1.4 Show understanding of the analysis, design, coding, testing and maintenance stages in the program development life cycle

- **Analysis**

- **Describe [1]**

- Feasibility study // Problem definition / investigation // Requirements etc.

- **Product [1]**

- The problem definition
 - Requirements specification // Client requirements
 - Documentation related to current system (e.g. E-R diagram of current system, DFD of current system, feasibility study)

- **Design**

- **Describe [1]**

- To define the data structures and algorithms(of the solution).
- **Describe(further) [3]** *(to make decisions about)*
 - Data structures // data dictionary // identifier tables // validation rules
 - Algorithms
 - Program structure / use of library routines / **module - team allocation**
 - User interface
 - Testing method / plan
 - Programming language // Development
- Coding
 - The code is written
- Testing
 - *Have its own section below*
- Maintenance
 - *Have its own section below*

12.2 Program Design

12.2.1 Use a structure chart to decompose a problem into sub-tasks and express the parameters passed between the various modules / procedures / functions which are part of the algorithm design

- Describe the purpose of a structure chart
- Construct a structure chart for a given problem
- Derive equivalent pseudocode from a structure chart
- **Purpose of structure chart [2]**

- module relationships / hierarchy / selection / repetition // how a problem is decomposed
- the parameters that are passed between the sub-tasks / modules // whether a module is a function or a procedure
- **Features that can be represented on a structure chart [2]**
 - Module names and parameters
 - Hierarchy / relationships
 - Selection
 - Repetition / Iteration
 - Sequence
- **Construct a structure chart for a given problem [5]**
 - Diagram with all boxes correctly labelled and positioned
 - Selection diamond (if have)
 - Iteration arrow (if have)
 - One mark per type of parameter e.g. BYREF, BYVALUE, BOOLEAN and the returned parameter
- **Meaning of diamond symbol [2]**
 - Means that a module calls either one of its sub-modules
 - One mark for reference to selection
 - One mark for naming all modules correctly
- **Meaning of curved arrow symbol [2]**
 - iteration / looping
 - naming all modules correctly in the correct sequence

12.2.2 Show understanding of the purpose of state-transition diagrams to document an algorithm
(analysed from DeepSeek)

- To visually represent the different states of a system and the transitions (changes) between them, typically triggered by events or conditions.
- Help modelling and understanding the behaviour of systems, such as software processes or finite state machines.

12.3 Program Testing and Maintenance

12.3.1 Show understanding of ways of exposing and avoiding faults in programs

- **Good programming practices [1]**

- Indentation / use of white space (*indicate the purpose of the variable*)
- Meaningful identifier name
- Capitalisation of keywords
- Use of comments
- Use of modular programming
- Use of local variables

12.3.2 Locate and identify the different types of errors

- syntax errors
- logic errors
- run-time errors

- **Syntax error [1]**

- Rules of programming language have not been followed

- **Logic error [1]**

- Where the program does not behave as expected / does not give the expected result / an error in the logic in the program

- **Run-time error [1]**

- The program executes an illegal instruction // performs an illegal operation that is trapped by the OS
- e.g.
 - Division by zero
 - Attempt to convert a non-numeric string to a number

try 9618_w24_qp_22 Q5(a), 9618_s24_qp_22 Q5(a)

12.3.3 Correct identified errors

You can do this.

12.3.4 Show understanding of the methods of testing available and select appropriate data for a given method

Including dry run, walkthrough, white-box, black-box, integration, alpha, beta, acceptance, stub

- **Dry run** (*essentially same to walkthrough, just different scales. You can directly use the description of walkthrough*)

- **Walkthrough [3]**

- Going through the program a line at a time
- ... using a trace table
- ... to record/check variable (values) as they change
- draw up test data, e.g. boundary, normal, abnormal data
- Error may be indicated when variable given an unexpected value
- Error may be indicated by an unexpected path through the program // Faults in the logic of the program can be detected

- **White box testing [2]**
 - choosing data/values
 - test every possible 'logic path' through the code
- **Black box testing [2]**
 - To test that the program does what it is supposed to do / to check that the results are as expected
 - to use known valid, boundary and erroneous values
- **Integration testing [2]**
 - Modules that have already been tested individually
 - are combined into a single (sub) program which is then tested as a whole
- **Alpha testing (*analysed from A2 paper*)**
 - Testing carried out by developers
 - Near the end of development // after integration before beta
 - To find errors not found in earlier testing // ensure ready for beta testing
- **Beta testing [3]**
 - Testing carried out by a small group of (potential) users
 - Users will check that the software works as required / works in the real world / doesn't contain errors
 - Users will feedback problems / suggestions for improvement
 - Problems / suggestions identified are addressed (before the problem is sold)
- **Acceptance (*analysed from textbook*)**
 - customer check if program meets their requirements and works as expected
 - generally part of the hand-over process
 - If successful, customer will sign off the software
- **Stub testing [2~3]**

- Test the program before all subroutines have been implemented
- A **simplified** version of the subroutine is written to **replace** each unsimplified modules
- that returns typical / expected value / known response

12.3.5 Show understanding of the need for a test strategy and test plan and their likely contents

- Stub testing
 - when a module is found erroneous but we want to continue testing
 - when we want to test when some subroutines are not written
- Describe a test plan [2]
 - Create a list of numbers that should generate **all of the** possible check digit values
 - ... and check each generated value as expected
 - ... by giving examples

12.3.6 Choose appropriate test data for a test plan

Including normal, abnormal and extreme/boundary

Type of test data	Explanation
Normal	Value within acceptable range
Abnormal	Value outside acceptable range
Extreme	Maximum acceptable // Minimum acceptable
Boundary	Maximum unacceptable // Minimum unacceptable // Maximum acceptable // Minimum acceptable

12.3.7 Show understanding of the need for continuing maintenance of a system and the differences between each type of maintenance

Including perfective, adaptive, corrective

- **Perfective [1]**

- To improve the performance of the program / to enhance the program functionality

- **Adaptive**

- **Describe [2]**

- Program code is modified
 - Following a change to the requirements
 - Change in relevant legislation
 - New technology available

- **Example [2]**

- New technologies available // changes made to library modules used
 - Changes in relevant legislation

- **Corrective [1]**

- When program does not function as intended / there is a bug

- **Preventive [1] (*not in syllabus*)**

- Improve maintainability or reliability

12.3.8 Analyse an existing program and make amendments to enhance functionality

try 9618_w22_qp_22 Q5

try 9618_s24_qp_23 Q5

13 Appendix

13.1 Generic Marking Principle

Generic Marking Principles

These general marking principles must be applied by all examiners when marking candidate answers. They should be applied alongside the specific content of the mark scheme or generic level descriptors for a question. Each question paper and mark scheme will also comply with these marking principles.

GENERIC MARKING PRINCIPLE 1:

Marks must be awarded in line with:

- the specific content of the mark scheme or the generic level descriptors for the question
- the specific skills defined in the mark scheme or in the generic level descriptors for the question
- the standard of response required by a candidate as exemplified by the standardisation scripts.

GENERIC MARKING PRINCIPLE 2:

Marks awarded are always **whole marks** (not half marks, or other fractions).

GENERIC MARKING PRINCIPLE 3:

Marks must be awarded **positively**:

- marks are awarded for correct/valid answers, as defined in the mark scheme. However, credit is given for valid answers which go beyond the scope of the syllabus and mark scheme, referring to your Team Leader as appropriate
- marks are awarded when candidates clearly demonstrate what they know and can do
- marks are not deducted for errors
- marks are not deducted for omissions
- answers should only be judged on the quality of spelling, punctuation and grammar when these features are specifically assessed by the question as indicated by the mark scheme. The meaning, however, should be unambiguous.

GENERIC MARKING PRINCIPLE 4:

Rules must be applied consistently, e.g. in situations where candidates have not followed instructions or in the application of generic level descriptors.

GENERIC MARKING PRINCIPLE 5:

Marks should be awarded using the full range of marks defined in the mark scheme for the question (however; the use of the full mark range may be limited according to the quality of the candidate responses seen).

GENERIC MARKING PRINCIPLE 6:

Marks awarded are based solely on the requirements as defined in the mark scheme. Marks should not be awarded with grade thresholds or grade descriptors in mind.

13.2 Insert for paper 2

Note: content of insert may change depending on the exam

An error will be generated if a function call is not properly formed or if the parameters are of an incorrect type or an incorrect value.

String and character functions

- A string of length 1 may be considered to be either of type CHAR or STRING
- A CHAR may be assigned to, or concatenated with, a STRING
- A STRING of length greater than 1 cannot be assigned to a CHAR

LEFT(ThisString : STRING, x : INTEGER) RETURNS STRING

returns leftmost x characters from ThisString

Example: LEFT("ABCDEFGH", 3) returns "ABC"

RIGHT(ThisString : STRING, x : INTEGER) RETURNS STRING

returns rightmost x characters from ThisString

Example: RIGHT("ABCDEFGH", 3) returns "FGH"

MID(ThisString : STRING, x : INTEGER, y : INTEGER) RETURNS STRING

returns a string of length y starting at position x from ThisString

Example: MID("ABCDEFGH", 2, 3) returns string "BCD"

LENGTH(ThisString : STRING) RETURNS INTEGER

returns the integer value representing the length of ThisString

Example: LENGTH("Happy Days") returns 10

TO_UPPER(x : <datatype>) RETURNS <datatype>

<datatype> may be CHAR or STRING

returns an object of type <datatype> formed by converting all characters of x to upper case.

Examples:

- TO_UPPER("Error 803") returns "ERROR 803"
- TO_UPPER('a') returns 'A'

TO_LOWER(x : <datatype>) RETURNS <datatype>

<datatype> may be CHAR or STRING

returns an object of type <datatype> formed by converting all characters of x to lower case.

Examples:

- TO_LOWER("JIM 803") returns "jim 803"
- TO_LOWER('W') returns 'w'

NUM_TO_STR(x : <datatype1>) RETURNS <datatype2>

returns a string representation of a numeric value.

<datatype1> may be REAL or INTEGER, <datatype2> may be CHAR or STRING

Example: NUM_TO_STR(87.5) returns "87.5"

If x is a negative value, the returned value will be a string beginning with the '-' character.

STR_TO_NUM(x : <datatype1>) RETURNS <datatype2>

returns a numeric representation of a string.

<datatype1> may be CHAR or STRING, <datatype2> may be REAL or INTEGER

Example: STR_TO_NUM("23.45") returns 23.45

If the string begins with the '-' character, the returned value will be negative.

IS_NUM(ThisString : <datatype>) RETURNS BOOLEAN

returns TRUE if ThisString represents a valid numeric value.

<datatype> may be CHAR or STRING

Example: IS_NUM("-12.36") returns TRUE

ASC(ThisChar : CHAR) RETURNS INTEGER

returns an integer value (the ASCII value) of character ThisChar

Example: ASC('A') returns 65, ASC('B') returns 66

CHR(x : INTEGER) RETURNS CHAR

returns the character whose integer value (the ASCII value) is x

Example: CHR(65) returns 'A', CHR(66) returns 'B'

Numeric functions

INT(x : REAL) RETURNS INTEGER

returns the integer part of x

Example: INT(27.5415) returns 27

RAND(x : INTEGER) RETURNS REAL

returns a real number in the range 0 to x (not inclusive of x).

Example: RAND(87) could return 35.430729

Date functions

Date format is assumed to be DD/MM/YYYY unless otherwise stated.

DAY(ThisDate : DATE) RETURNS INTEGER

returns the day number from ThisDate

Example: DAY(04/10/2003) returns 4

MONTH(ThisDate : DATE) RETURNS INTEGER

returns the month number from ThisDate

Example: MONTH(04/10/2003) returns 10

YEAR(ThisDate : DATE) RETURNS INTEGER

returns the year number from ThisDate

Example: YEAR(04/10/2003) returns 2003

DAYINDEX(ThisDate : DATE) RETURNS INTEGER

returns the day index number from ThisDate where Sunday = 1, Monday = 2 etc.

Example: DAYINDEX(09/05/2023) returns 3

SETDATE(Day, Month, Year : INTEGER) RETURNS DATE

returns a value of type DATE with the value of <Day>/<Month>/<Year>

Example: SETDATE(26, 10, 2003) returns a date corresponding to 26/10/2003

TODAY() RETURNS DATE

returns a value of type DATE corresponding to the current date.

Text file functions

EOF(FileName : STRING) RETURNS BOOLEAN
returns TRUE if there are no more lines to be read from file FileName will generate an error if the file is not already open in READ mode.

Operators

An error will be generated if an operator is used with a value or values of an incorrect type.

&	concatenates (joins) two strings. Example: "Summer" & " " & "Pudding" evaluates to "Summer Pudding" may also be used to concatenate a CHAR with a STRING
AND	performs a logical AND on two Boolean values. Example: TRUE AND FALSE evaluates to FALSE
OR	performs a logical OR on two Boolean values. Example: TRUE OR FALSE evaluates to TRUE
NOT	performs a logical NOT on a Boolean value. Example: NOT TRUE evaluates to FALSE
MOD	finds the remainder when one number is divided by another. Example: 10 MOD 3 evaluates to 1
DIV	finds the quotient when one number is divided by another. Example 10 DIV 3 evaluates to 3

Comparison operators

=	used to compare two items of the same type. evaluates to TRUE if the condition is true, otherwise evaluates to FALSE
>	Notes: • may be used to compare types REAL and INTEGER • may be used to compare types CHAR and STRING • case sensitive when used to compare types CHAR and/or STRING • cannot be used to compare two records Examples: • "Program" = "program" evaluates to FALSE • Count = 4 evaluates to TRUE when Count contains the value 4
<	
>=	
<=	
<>	

13.3 Pseudocode symbols and keywords

Cambridge International AS & A Level Computer Science 9618 Pseudocode Guide for Teachers for 2026.

Index of symbols and keywords

- , 16	GETRECORD , 27
← , 10	IF , 19
* , 16	INHERITS , 29
/ , 16	INPUT , 16
// , 7	INT , 18
+ , 16	INTEGER , 8
< , 17	LCASE , 18
<= , 17	LENGTH , 17
<> , 17	MID , 18
= , 17	MOD , 16
> , 17	NEXT , 21
>= , 17	NEW , 29
^ (caret) , 13	NOT , 17
& , 18	OPENFILE , 26
AND , 17	OR , 17
APPEND , 26	OTHERWISE , 20
ARRAY , 11	OUTPUT , 16
BOOLEAN , 8	PROCEDURE , 23
BYREF , 25	PRIVATE , 29
BYVAL , 25	PUBLIC , 29
CALL , 23	PUTRECORD , 28
CASE OF , 20	RAND , 18
CHAR , 8	RANDOM (files) , 27
CLASS , 29	READ , 26
CLOSEFILE , 26	READFILE , 26
CONSTANT , 9	REAL , 8
DATE , 8	REPEAT , 21
DECLARE , 9	RETURN , 24
DIV , 16	RETURNS , 24
ELSE , 19	RIGHT , 17
ENDCASE , 20	SEEK , 27
ENDCLASS , 29	STEP , 21
ENDFUNCTION , 24	STRING , 8
ENDIF , 19	SUPER , 29
ENDPROCEDURE , 23	THEN , 19
ENDTYPE , 14	TRUE , 8
ENDWHILE , 22	TYPE , 13
EOF , 26	UCASE , 18
FALSE , 8	UNTIL , 21
FOR ... TO , 21	WHILE , 22
FOR (file handling) , 26	WRITE , 26
FUNCTION , 24	WRITEFILE , 26

13.4 Further resources

For further resources e.g. past papers, syllabus, topical question, examiner reports, refer to https://drive.google.com/drive/folders/1C1RXjXPq_yIN9hchMCC6GD8uYkCHQmTK?usp=drive_link

13.5 Epilogue

As you can see, this document is far from being perfect. If you are comfortable working with *Pages*, you can join this line of work by contacting my email: darkgoatlord@gmail.com.