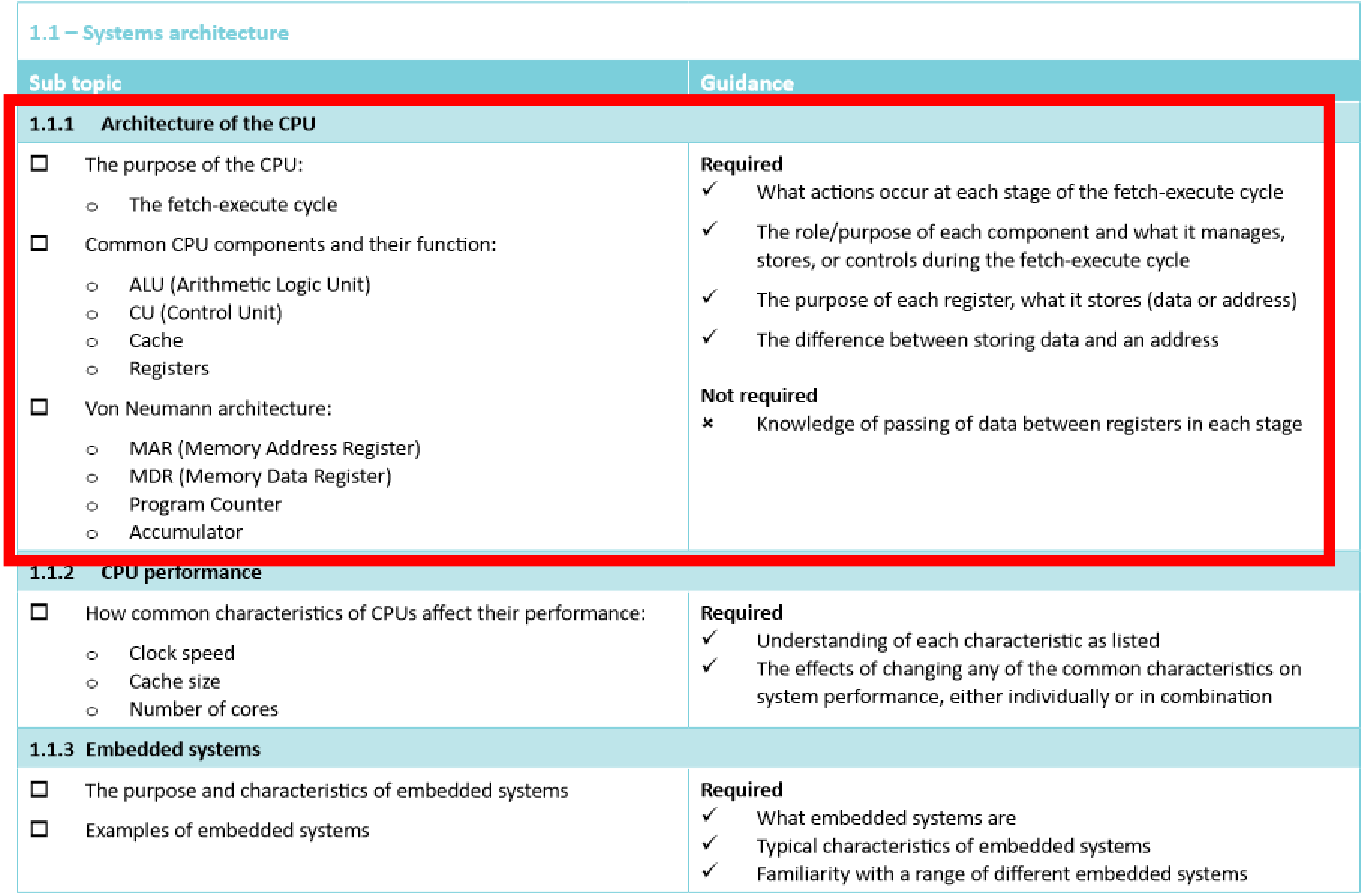
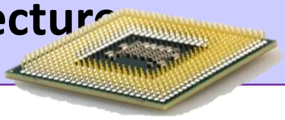
J277 - 1.1 Systems architecture



1.1 – Systems architecture

Sub topic	Guidance
1.1.1 Architecture of the CPU	
☐ The purpose of the CPU: ○ The fetch-execute cycle ☐ Common CPU components and their function: ○ ALU (Arithmetic Logic Unit) ○ CU (Control Unit) ○ Cache ○ Registers ☐ Von Neumann architecture: ○ MAR (Memory Address Register) ○ MDR (Memory Data Register) ○ Program Counter ○ Accumulator	**Required** ✓ What actions occur at each stage of the fetch-execute cycle ✓ The role/purpose of each component and what it manages, stores, or controls during the fetch-execute cycle ✓ The purpose of each register, what it stores (data or address) ✓ The difference between storing data and an address **Not required** ✗ Knowledge of passing of data between registers in each stage
1.1.2 CPU performance	
☐ How common characteristics of CPUs affect their performance: ○ Clock speed ○ Cache size ○ Number of cores	**Required** ✓ Understanding of each characteristic as listed ✓ The effects of changing any of the common characteristics on system performance, either individually or in combination
1.1.3 Embedded systems	
☐ The purpose and characteristics of embedded systems ☐ Examples of embedded systems	**Required** ✓ What embedded systems are ✓ Typical characteristics of embedded systems ✓ Familiarity with a range of different embedded systems

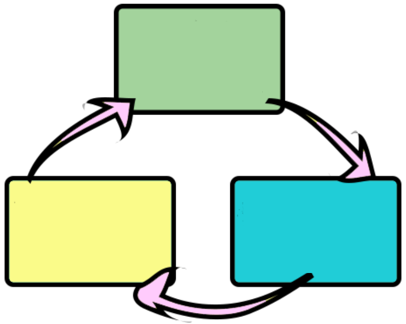
J277 - 1.1 Systems architecture

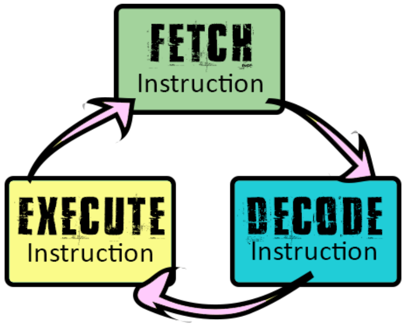


The Central Processing Unit (CPU)

This is where a computer processes all data and instructions.

The CPU has a small amount of memory called registers.





Registers:	**Memory Address Register (MAR)**	**Memory Data Register (MDR)**	**Program Counter (PC)**	**Accumulator (ACC)**
Purpose:	Stores the location in memory to be used by the MDR.	Holds the actual data or instruction fetched from, or to be written to the memory.	Holds the address of the next instruction.	Holds the results of calculations from the ALU.

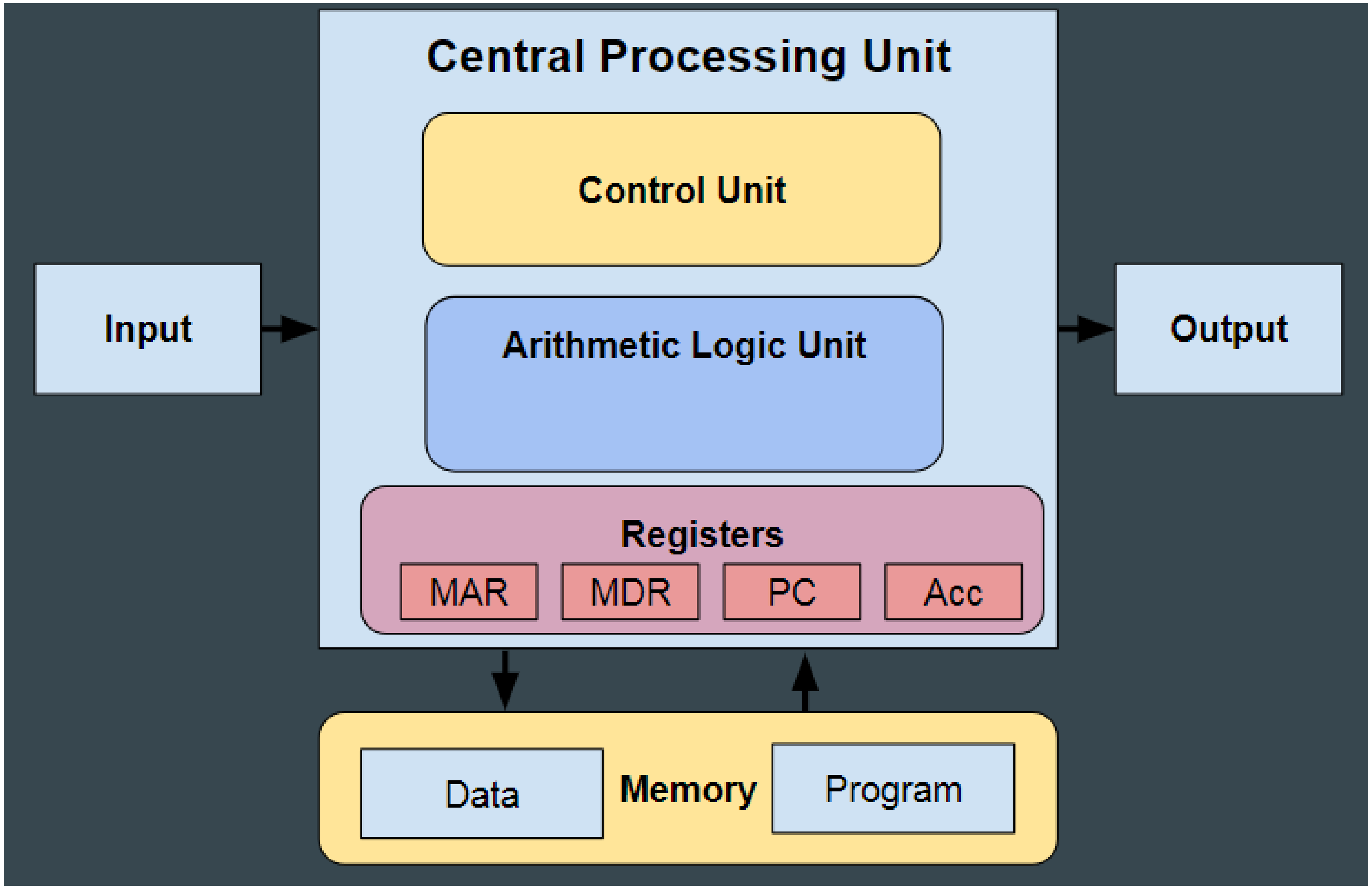
The CPU also has these components:

Arithmetic logic unit (ALU)	**Control Unit**	**Cache**
Does calculations including addition, subtraction, multiplication and division. Also performs binary shifts and logical decisions.	Controls the flow of data in and out of the CPU. Manages fetching, decoding and executing of instructions.	This memory provides fast access to regularly used instructions and data without having to go to the main memory (RAM).

J277 - 1.1 Systems architecture

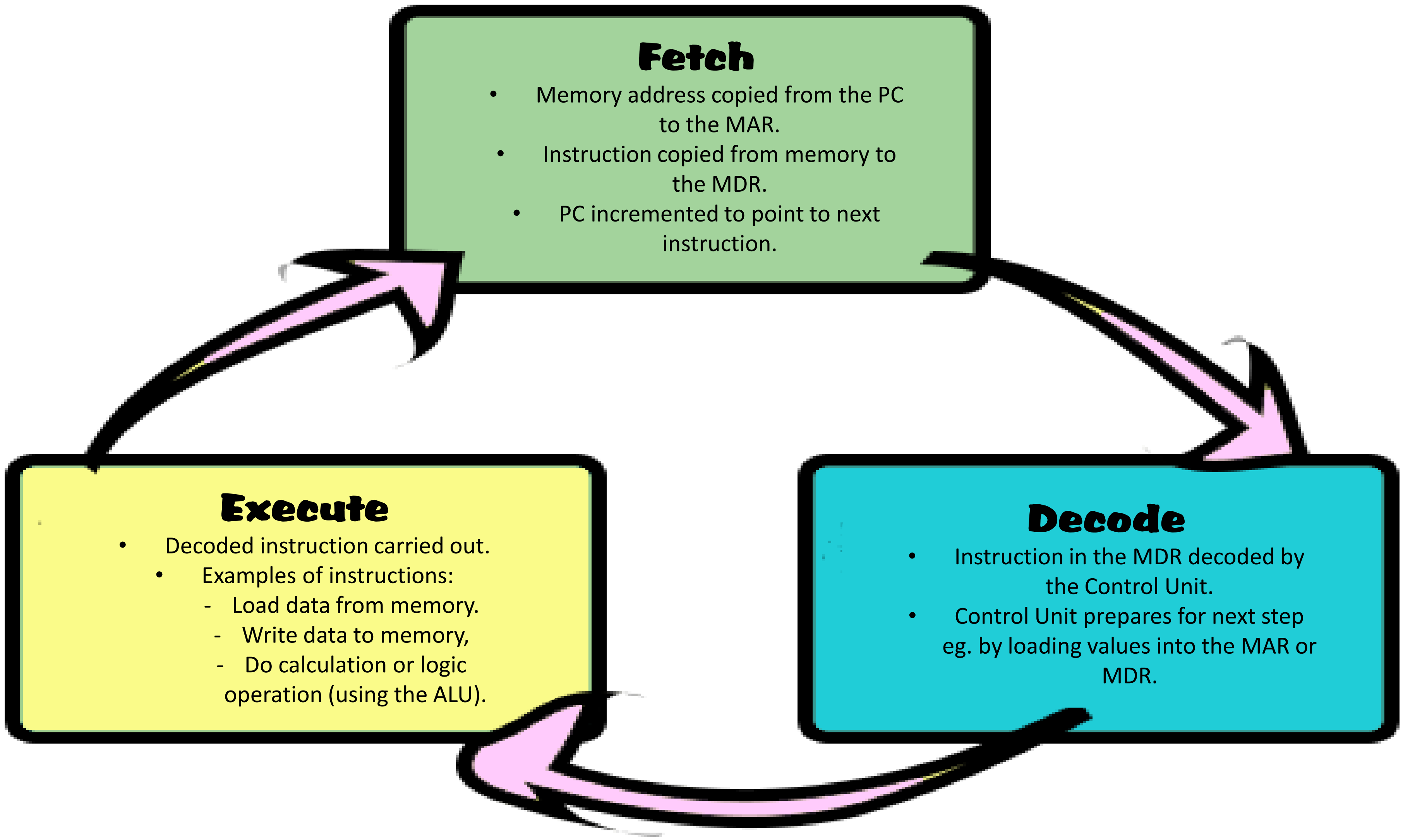
How the CPU works: the Von Neumann architecture

In the Von Neumann architecture, data and instructions are both stored in the same memory.



How the CPU works: Fetch-Execute Cycle

The Fetch-Execute cycle repeats continuously while the computer is running.



Fetch

- Memory address copied from the PC to the MAR.
- Instruction copied from memory to the MDR.
- PC incremented to point to next instruction.

Decode

- Instruction in the MDR decoded by the Control Unit.
- Control Unit prepares for next step eg. by loading values into the MAR or MDR.

Execute

- Decoded instruction carried out.
- Examples of instructions:
  - Load data from memory.
  - Write data to memory,
  - Do calculation or logic operation (using the ALU).

J277 - 1.1 Systems architecture

1.1 – Systems architecture

Sub topic	Guidance
1.1.1 Architecture of the CPU	
☐ The purpose of the CPU: ○ The fetch-execute cycle ☐ Common CPU components and their function: ○ ALU (Arithmetic Logic Unit) ○ CU (Control Unit) ○ Cache ○ Registers ☐ Von Neumann architecture: ○ MAR (Memory Address Register) ○ MDR (Memory Data Register) ○ Program Counter ○ Accumulator	**Required** ✓ What actions occur at each stage of the fetch-execute cycle ✓ The role/purpose of each component and what it manages, stores, or controls during the fetch-execute cycle ✓ The purpose of each register, what it stores (data or address) ✓ The difference between storing data and an address **Not required** ✗ Knowledge of passing of data between registers in each stage
1.1.2 CPU performance	
☐ How common characteristics of CPUs affect their performance: ○ Clock speed ○ Cache size ○ Number of cores	**Required** ✓ Understanding of each characteristic as listed ✓ The effects of changing any of the common characteristics on system performance, either individually or in combination
1.1.3 Embedded systems	
☐ The purpose and characteristics of embedded systems ☐ Examples of embedded systems	**Required** ✓ What embedded systems are ✓ Typical characteristics of embedded systems ✓ Familiarity with a range of different embedded systems

J277 - 1.1 Systems architecture

Factors affecting the speed of the CPU

Generally, CPUs with more cores, higher clock speeds and larger caches will have better performance, but will cost more.

Clock speed

The number of instructions a single processor core can carry out per second, measured in hertz.

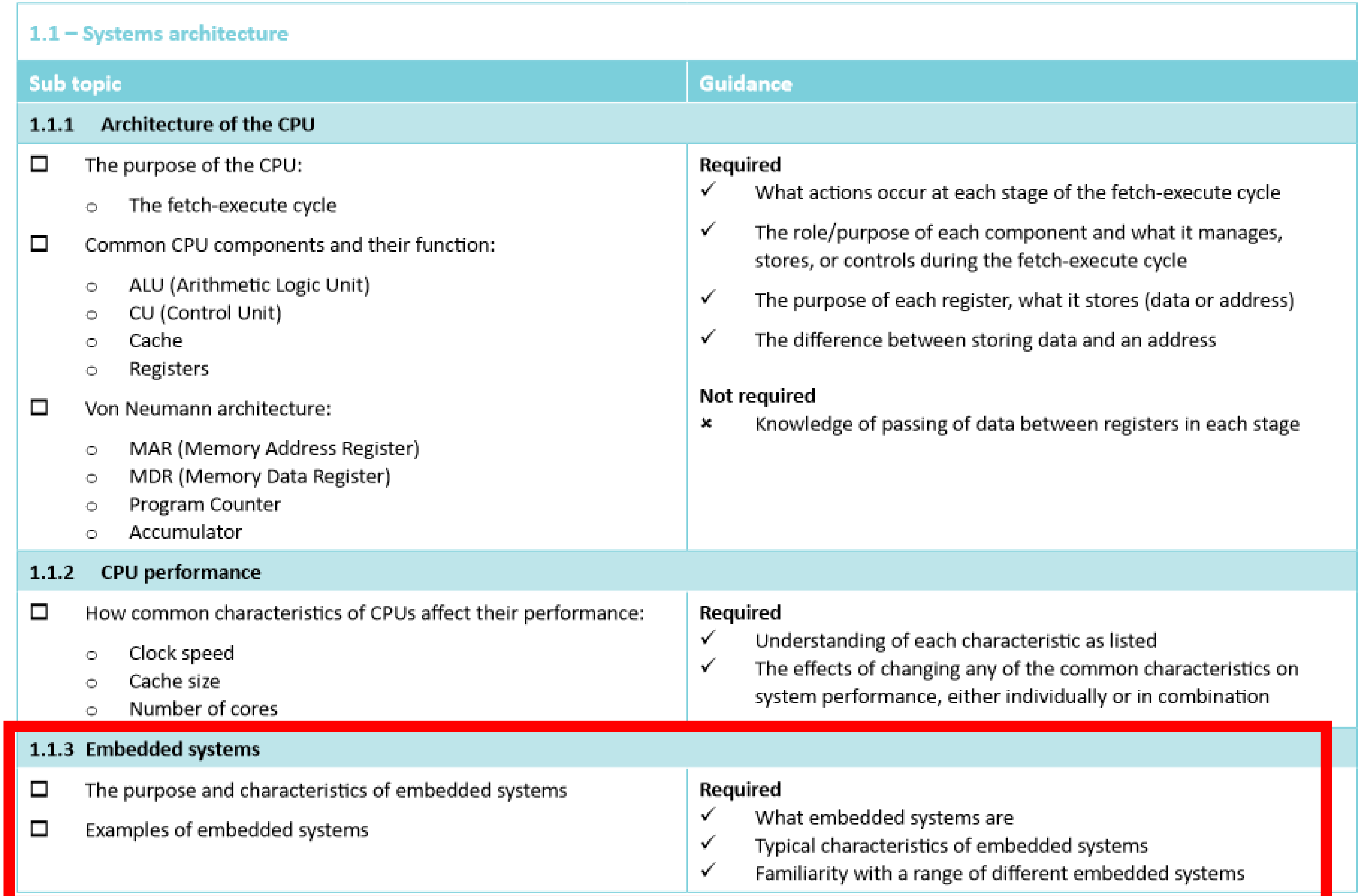
Number of cores

Each core processes data independently, so more cores means more instructions carries out per second.

Cache size

Cache is a small amount of memory in the CPU used to store frequently used programs so that they can be fetched quicker (rather than from RAM).

J277 - 1.1 Systems architecture



1.1 – Systems architecture

Sub topic	Guidance
1.1.1 Architecture of the CPU	
☐ The purpose of the CPU: ○ The fetch-execute cycle ☐ Common CPU components and their function: ○ ALU (Arithmetic Logic Unit) ○ CU (Control Unit) ○ Cache ○ Registers ☐ Von Neumann architecture: ○ MAR (Memory Address Register) ○ MDR (Memory Data Register) ○ Program Counter ○ Accumulator	**Required** ✓ What actions occur at each stage of the fetch-execute cycle ✓ The role/purpose of each component and what it manages, stores, or controls during the fetch-execute cycle ✓ The purpose of each register, what it stores (data or address) ✓ The difference between storing data and an address **Not required** ✗ Knowledge of passing of data between registers in each stage
1.1.2 CPU performance	
☐ How common characteristics of CPUs affect their performance: ○ Clock speed ○ Cache size ○ Number of cores	**Required** ✓ Understanding of each characteristic as listed ✓ The effects of changing any of the common characteristics on system performance, either individually or in combination
1.1.3 Embedded systems	
☐ The purpose and characteristics of embedded systems ☐ Examples of embedded systems	**Required** ✓ What embedded systems are ✓ Typical characteristics of embedded systems ✓ Familiarity with a range of different embedded systems

J277 - 1.1 Systems architecture

Embedded systems

Embedded systems are:

Computer systems that are built into other devices, with a dedicated function within a larger control system.

Examples of embedded systems:

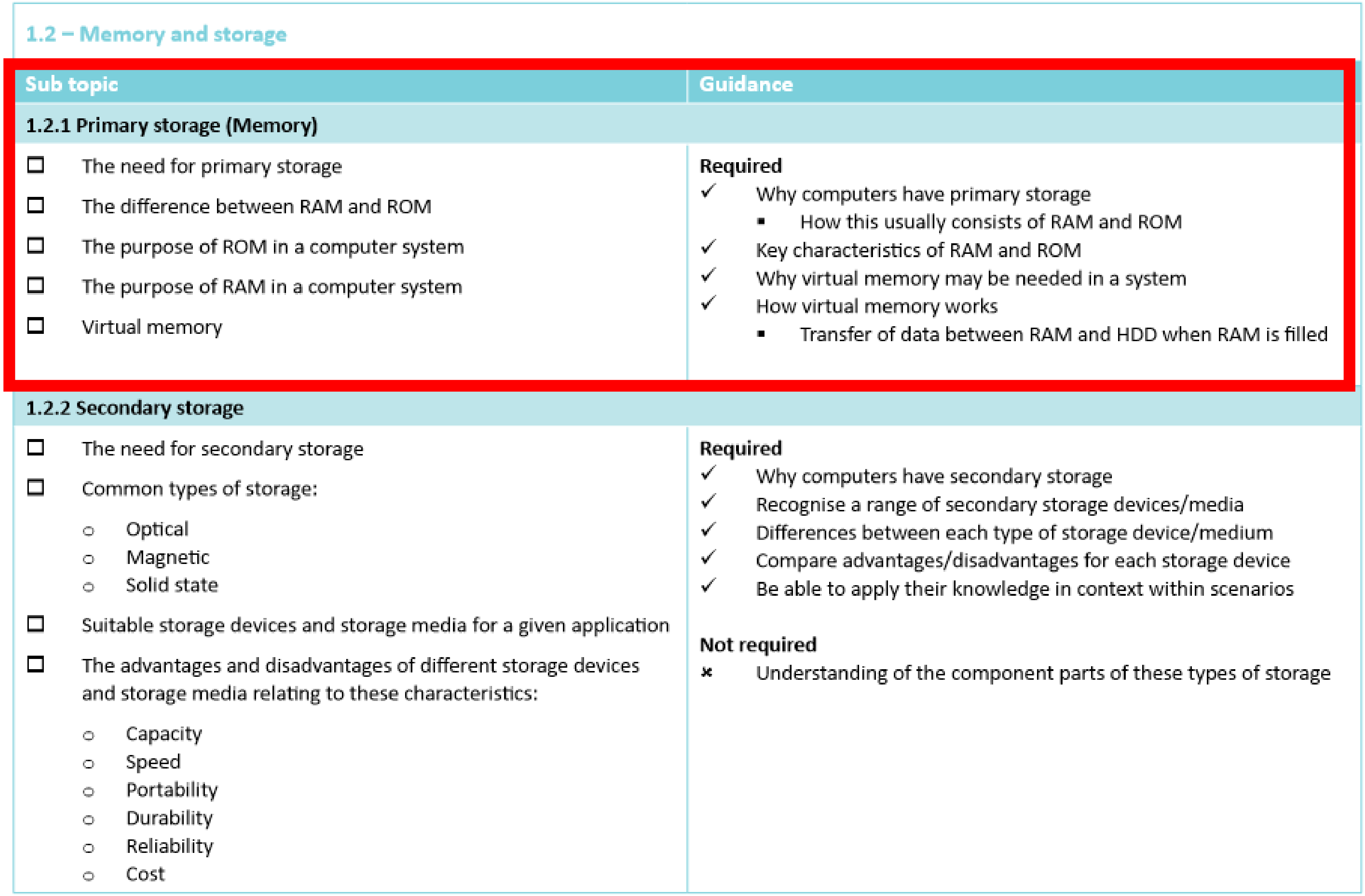
They're usually **easier** to design, **cheaper** to produce, and more **efficient** at their task than general purpose systems.

Washing machine
User can select a wash program.
A sequence of fill, turn, drain cycles wash the clothes.

Sat Nav
User can select a route.
A sequence of directions are outputs driving to a destination.

Digital watches
User can select the app, the process will be completed on the app and then the output will be displayed.

J277 - 1.2 Memory & Storage



1.2 – Memory and storage

Sub topic	Guidance
1.2.1 Primary storage (Memory)	
☐ The need for primary storage ☐ The difference between RAM and ROM ☐ The purpose of ROM in a computer system ☐ The purpose of RAM in a computer system ☐ Virtual memory	**Required** ✓ Why computers have primary storage ▪ How this usually consists of RAM and ROM ✓ Key characteristics of RAM and ROM ✓ Why virtual memory may be needed in a system ✓ How virtual memory works ▪ Transfer of data between RAM and HDD when RAM is filled
1.2.2 Secondary storage	
☐ The need for secondary storage ☐ Common types of storage: ○ Optical ○ Magnetic ○ Solid state ☐ Suitable storage devices and storage media for a given application ☐ The advantages and disadvantages of different storage devices and storage media relating to these characteristics: ○ Capacity ○ Speed ○ Portability ○ Durability ○ Reliability ○ Cost	**Required** ✓ Why computers have secondary storage ✓ Recognise a range of secondary storage devices/media ✓ Differences between each type of storage device/medium ✓ Compare advantages/disadvantages for each storage device ✓ Be able to apply their knowledge in context within scenarios **Not required** ✗ Understanding of the component parts of these types of storage

J277 - 1.2 Memory & Storage

Random Access Memory (RAM)

Name

Random Access Memory.

Purpose

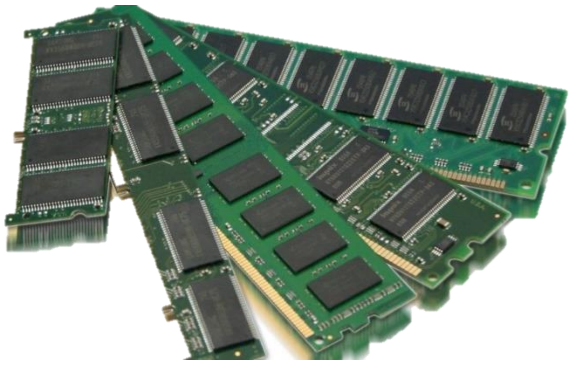
A temporary store of instructions and data in use by the CPU for currently executing programs. Programs and data are loaded from the hard disk/solid state storage to RAM for processing.

Volite memory.

Can be read and written to.

Programs and files are copied here from secondary storage while in use.

Slower than the CPU Cache, but faster than secondary storage.



J277 - 1.2 Memory & Storage

Read Only Memory (ROM)

Name

Read Only Memory.

Purpose

Holds the first instructions to execute when the computer is first turned on. Also known as the ‘BIOS’.

- Non-Volite memory.
- Can only be read from, not written to.
- Programs and files are copied here from secondary storage while in use.
- Contains BIOS (Basic Input Output System) – instructions needed for the computer to boot up.

J277 - 1.2 Memory & Storage

Virtual Memory

Name

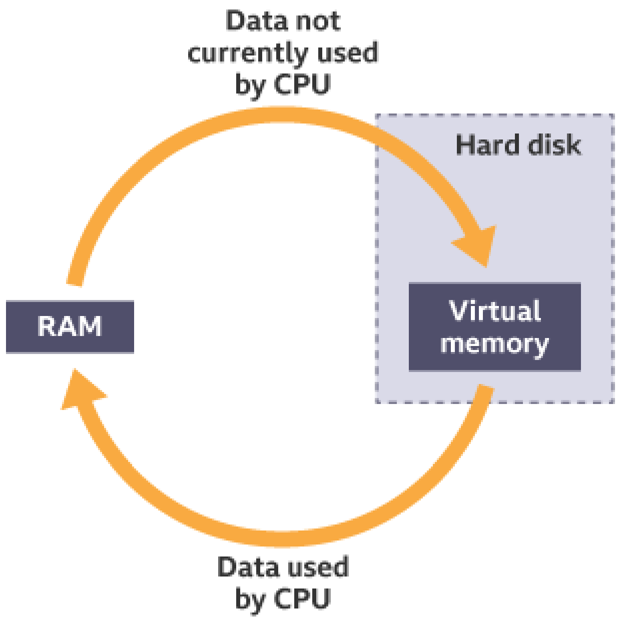
Virtual Memory.

Purpose

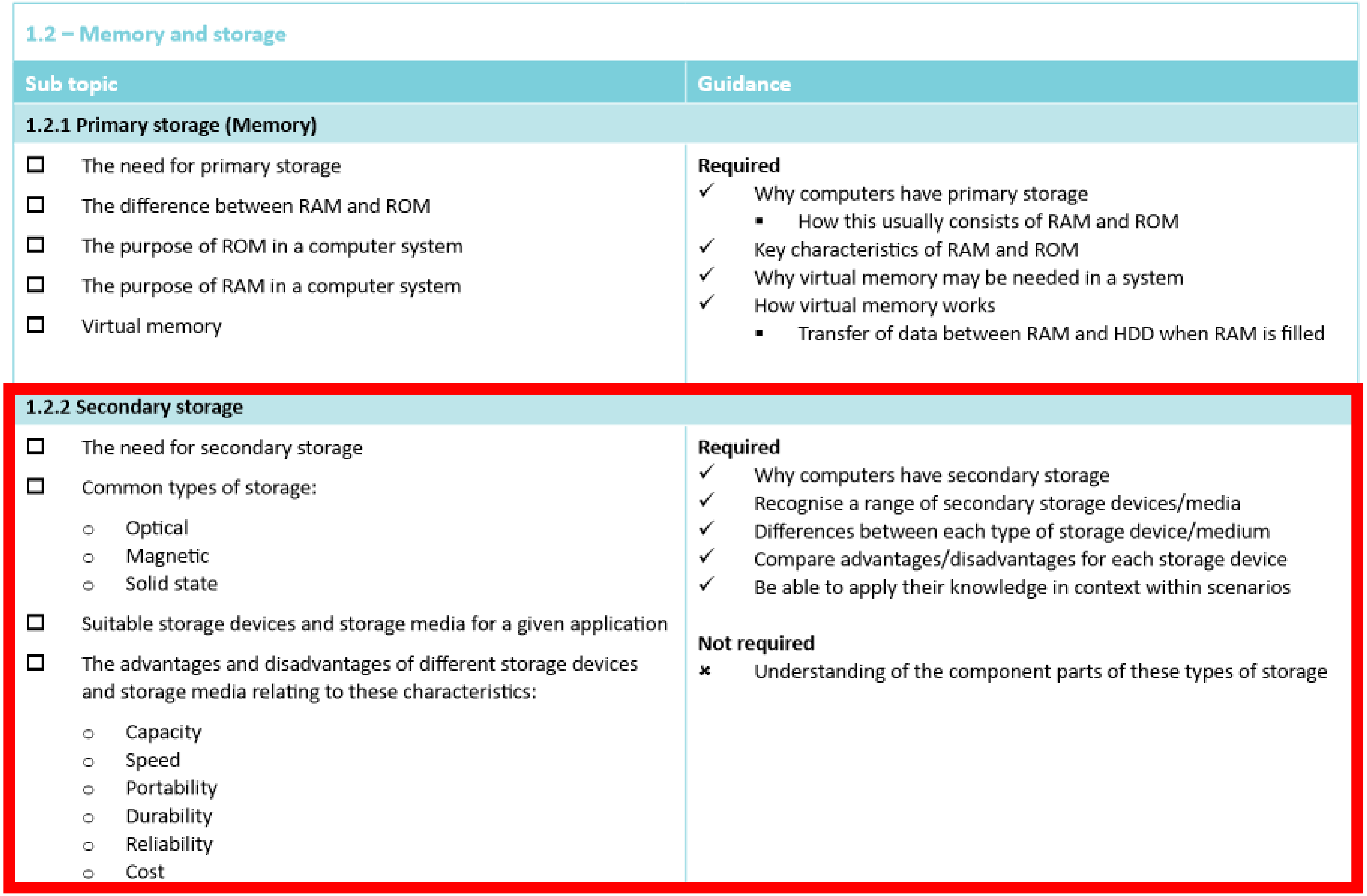
This is where part of the hard disk as if it is memory for programs that are running, when the RAM is full.

Issue

Computer instructions can only be executed from the cache and RAM. As a result, they must be transported back from virtual memory before they can be used. This makes the computer slower as it has to continually move instructions to the RAM from the virtual memory.



J277 - 1.2 Memory & Storage



1.2 – Memory and storage

Sub topic	Guidance
1.2.1 Primary storage (Memory)	
☐ The need for primary storage ☐ The difference between RAM and ROM ☐ The purpose of ROM in a computer system ☐ The purpose of RAM in a computer system ☐ Virtual memory	**Required** ✓ Why computers have primary storage ▪ How this usually consists of RAM and ROM ✓ Key characteristics of RAM and ROM ✓ Why virtual memory may be needed in a system ✓ How virtual memory works ▪ Transfer of data between RAM and HDD when RAM is filled
1.2.2 Secondary storage	
☐ The need for secondary storage ☐ Common types of storage: ○ Optical ○ Magnetic ○ Solid state ☐ Suitable storage devices and storage media for a given application ☐ The advantages and disadvantages of different storage devices and storage media relating to these characteristics: ○ Capacity ○ Speed ○ Portability ○ Durability ○ Reliability ○ Cost	**Required** ✓ Why computers have secondary storage ✓ Recognise a range of secondary storage devices/media ✓ Differences between each type of storage device/medium ✓ Compare advantages/disadvantages for each storage device ✓ Be able to apply their knowledge in context within scenarios **Not required** ✗ Understanding of the component parts of these types of storage

J277 - 1.2 Memory & Storage

The need for secondary storage

Because main memory (RAM) is volatile, any data or programs currently being stored there will disappear once the power is lost i.e. the computer is switched off.

So secondary storage is used to retain a copy of programs and data that need to be kept long term.

Two types of Internal Storage

1. **Hard Disk Drives (HDDs)**
- Has moving parts.
- Stores data magnetically on metal disks.
- Can be noisy.

2. **Solid State Drives (SSDs)**
- No moving parts.
- Uses flash memory for faster read/write times.
- Usually quiet/silent.

Four types of External Storage

1. **Flash drives & Memory cards**
Solid state storage used to expand the capacity of small devices.

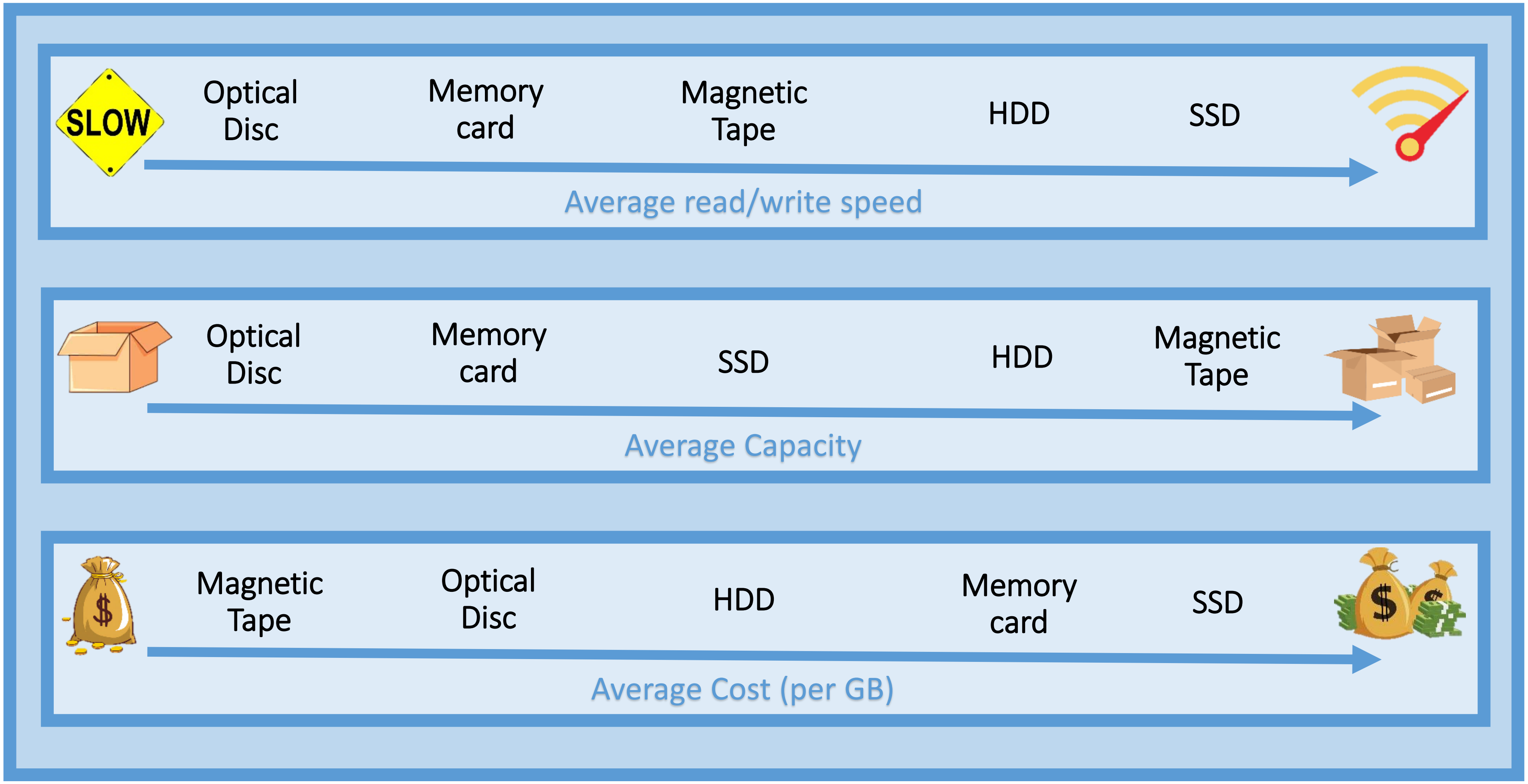
2. **Optical Discs**
Eg. CDs. Can be read-only, write-once or rewritable.

3. **Magnetic tape**
Used by organisations to store huge amounts of data.

4. **External HDDs & SSDs**
Portable versions of internal storage. Often used for back ups.

J277 - 1.2 Memory & Storage

Comparing Storage Types – Characteristics



Comparing Storage Types – Characteristics

	Internal HDD	Internal SSD	Memory card	Optical Disc	Magnetic tape
Portability	**Low**	**Low**	**High**	**High**	**High**
Durability and Reliability	•Damaged by impacts. •Long read/write life.	•Shock resistant. •Limited rewrites.	•Shock resistant. •Limited rewrites.	•Easily scratched. •Limited rewrites. •Suitable long term storage.	•Damaged by impacts, heat and magnets. •Suitable long term storage.

J277 - 1.2 Memory & Storage

Cloud storage

What is Cloud storage?

Storing programs and data on remote hard drives, accessed over the internet when needed.

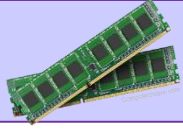
Advantages of cloud storage

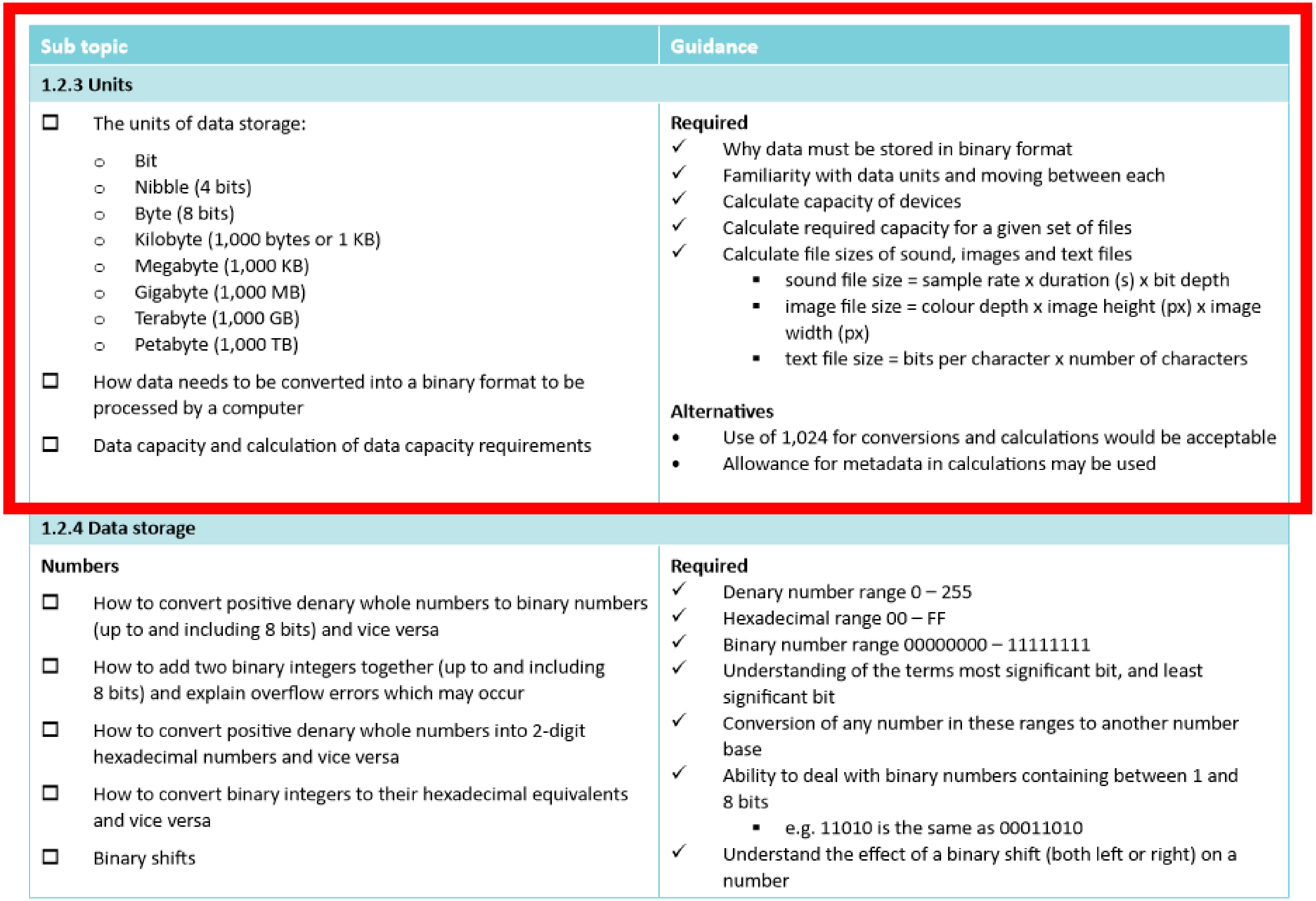
1. *Space is not taken up on local drives meaning more space is available overall.*
2. *Files can be accessed anywhere, from any device connected to the internet.*
3. *Backup and version history of files is usually kept automatically.*
4. *Collaboration on files is easy.*
5. *High quality images can be kept on remote servers, with lower quality images available on low storage devices.*

Disadvantages of cloud storage

1. *Monthly/annual cost.*
2. *Potential security risks.*
3. *Relying on a third party to look after your data.*
4. *Incompatibility between storage and applications.*

J277 - 1.2 Memory & Storage





Sub topic	Guidance
1.2.3 Units	
☐ The units of data storage: ○ Bit ○ Nibble (4 bits) ○ Byte (8 bits) ○ Kilobyte (1,000 bytes or 1 KB) ○ Megabyte (1,000 KB) ○ Gigabyte (1,000 MB) ○ Terabyte (1,000 GB) ○ Petabyte (1,000 TB) ☐ How data needs to be converted into a binary format to be processed by a computer ☐ Data capacity and calculation of data capacity requirements	**Required** ✓ Why data must be stored in binary format ✓ Familiarity with data units and moving between each ✓ Calculate capacity of devices ✓ Calculate required capacity for a given set of files ✓ Calculate file sizes of sound, images and text files ▪ sound file size = sample rate x duration (s) x bit depth ▪ image file size = colour depth x image height (px) x image width (px) ▪ text file size = bits per character x number of characters **Alternatives** • Use of 1,024 for conversions and calculations would be acceptable • Allowance for metadata in calculations may be used
1.2.4 Data storage	
Numbers ☐ How to convert positive denary whole numbers to binary numbers (up to and including 8 bits) and vice versa ☐ How to add two binary integers together (up to and including 8 bits) and explain overflow errors which may occur ☐ How to convert positive denary whole numbers into 2-digit hexadecimal numbers and vice versa ☐ How to convert binary integers to their hexadecimal equivalents and vice versa ☐ Binary shifts	**Required** ✓ Denary number range 0 – 255 ✓ Hexadecimal range 00 – FF ✓ Binary number range 00000000 – 11111111 ✓ Understanding of the terms most significant bit, and least significant bit ✓ Conversion of any number in these ranges to another number base ✓ Ability to deal with binary numbers containing between 1 and 8 bits ▪ e.g. 11010 is the same as 00011010 ✓ Understand the effect of a binary shift (both left or right) on a number

J277 - 1.2 Memory & Storage

Bit, nibble, byte, kilobyte & megabyte

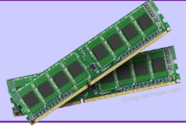
Traditionally, each unit is defined to be 1024 times bigger than the previous one!

Explain how binary data is used to represent numbers in a computer.

- Computers can only store and process binary data.
- They use 1s and 0s to represent the flow of electricity – A 1 shows that electricity is flowing and 0 shows that it isn’t.
- Each 1 or 0 in binary data is a **bit** (**b**inary dig**it**).

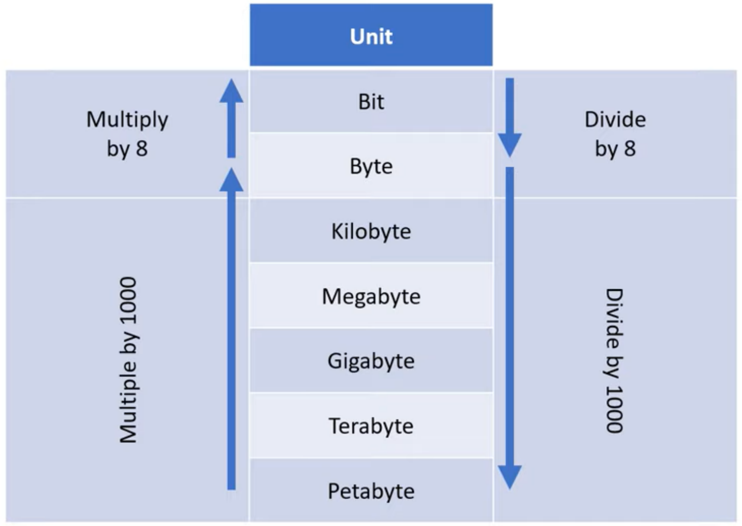
Name	**Size**
Bit	A single binary digit (0 or 1)
Nibble	4 bits
Byte (B)	8 bits
Kilobyte (KB)	1000 bytes
Megabyte (MB)	1000 kilobytes
Gigabyte (GB)	1000 megabytes
Terabyte (TB)	1000 gigabytes
Petabyte (PB)	1000 terabytes

J277 - 1.2 Memory & Storage



Units of data storage

Unit conversions:



Q1. Jenny has 700 high resolution 10Mb photographs.
What size memory card in GB will she need as a minimum to store these images?

7

Q2. An SD card stores films in a compressed format. Each film is 6.7GB. How many films can a 64GB card store?

9

Q3. A CCTV system records video at 180MB per second. How much storage is required in GB for 90 second of footage?

16.2

J277 - 1.2 Memory & Storage

Sub topic	Guidance
1.2.3 Units	
☐ The units of data storage: ○ Bit ○ Nibble (4 bits) ○ Byte (8 bits) ○ Kilobyte (1,000 bytes or 1 KB) ○ Megabyte (1,000 KB) ○ Gigabyte (1,000 MB) ○ Terabyte (1,000 GB) ○ Petabyte (1,000 TB) ☐ How data needs to be converted into a binary format to be processed by a computer ☐ Data capacity and calculation of data capacity requirements	**Required** ✓ Why data must be stored in binary format ✓ Familiarity with data units and moving between each ✓ Calculate capacity of devices ✓ Calculate required capacity for a given set of files ✓ Calculate file sizes of sound, images and text files ▪ sound file size = sample rate x duration (s) x bit depth ▪ image file size = colour depth x image height (px) x image width (px) ▪ text file size = bits per character x number of characters **Alternatives** • Use of 1,024 for conversions and calculations would be acceptable • Allowance for metadata in calculations may be used
1.2.4 Data storage	
Numbers ☐ How to convert positive denary whole numbers to binary numbers (up to and including 8 bits) and vice versa ☐ How to add two binary integers together (up to and including 8 bits) and explain overflow errors which may occur ☐ How to convert positive denary whole numbers into 2-digit hexadecimal numbers and vice versa ☐ How to convert binary integers to their hexadecimal equivalents and vice versa ☐ Binary shifts	**Required** ✓ Denary number range 0 – 255 ✓ Hexadecimal range 00 – FF ✓ Binary number range 00000000 – 11111111 ✓ Understanding of the terms most significant bit, and least significant bit ✓ Conversion of any number in these ranges to another number base ✓ Ability to deal with binary numbers containing between 1 and 8 bits ▪ e.g. 11010 is the same as 00011010 ✓ Understand the effect of a binary shift (both left or right) on a number

How to convert positive denary whole numbers (0-255) into 8 bit binary numbers

Converting 183 into Binary involves the following steps:

- 1. Does 128 go into 183? Yes so put a 1 Under 128. This leaves 55.

128	64	32	16	8	4	2	1
1							

- 2. Does 64 go into 55? No so put a 0 under 64. This leaves 55 still.

128	64	32	16	8	4	2	1
1	0						

- 3. Does 32 go into 55? Yes so put a 1 under 32. The leaves 23.

128	64	32	16	8	4	2	1
1	0	1					

- 4. Does 16 go into 23?. Yes so put a 1 under 16. This leaves 7.

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

- 5. Does 8 go into 7? No so put a 0 under 8. This still leaves 7.

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

- 6. Does 4 go into 7? Yes so put a 1 under 4. This leaves 3.

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

- 7. Does 2 go into 3? Yes so put a 1 under 2. This leaves 1.

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

- 8. Does 1 go into 1? Yes so put a 1 under 1.

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

J277 - 1.2 Memory & Storage

Convert these denary numbers into their 8 bit binary equivalent:

1. 20
2. 54
3. 125
4. 2
5. 99
6. 47

1. 00010100
2. 00110110
3. 01111101
4. 00000010
5. 01100011
6. 00101111

J277 - 1.2 Memory & Storage

Adding two 8 bit binary integers and overflow errors.

Binary numbers can easily be added using a few addition rules. These are:

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 10
1 + 1 + 1 = 11

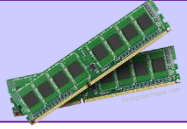
Examples

1. 00000101 + 00000110 = 00001011
2. 00001110 + 00000101 = 00010011
3. 00001001 + 00000011 = 00001100
4. 00000111 + 00001011 = 00010010
5. 00011011 + 00001010 = 00100001

Overflow errors can occur when adding binary numbers because:

There are not enough bits to hold the answer and a carry remains that cannot be stored.

J277 - 1.2 Memory & Storage



How to convert 8 bit binary numbers to positive denary whole numbers

The binary number 10110111 can be converted to Denary as:

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
1	0	1	1	0	1	1	1

If a ‘1’ is present in that column, then you need to add that number to the others…

This means you get: 128 + 0 + 32 + 16 + 0 + 4 + 2 + 1 = 183

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
1	0	1	1	0	1	1	1

J277 - 1.2 Memory & Storage

Convert these binary numbers into their denary equivalent:

1. 00000000 = 0
2. 00000001 = 1
3. 00001111 = 15
4. 00010010 = 18
5. 00101000 = 40
6. 01010101 = 85
7. 10000000 = 128
8. 11001100 = 204
9. 11100011 = 227
10. 11111111 = 255

J277 - 1.2 Memory & Storage

How to convert from binary to hexadecimal equivalents and vice versa.

To convert 4C from Hexadecimal into base 10:

STEP 1 – split the Hex up and find the binary representation:

4	C
0100	1100

STEP 2 – Put the binary number together, then convert to denary:

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

01001100 = 64+8+4 = 76

0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

J277 - 1.2 Memory & Storage

How to convert from hexadecimal to binary and vice versa.

We can convert from to Hexadecimal by using the following method:

e.g. Convert 167 into Hexadecimal

STEP 1 – Convert the number into binary:

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

STEP 2 – Split the binary number into two nibbles:

1010 0111

STEP 3 – Convert the two nibbles to Hex:

1010 = A 0111 = 7

Therefore the answer is = **A7**

0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

J277 - 1.2 Memory & Storage

Binary shifts

A left shift performs a: multiplication.

Question 2:

	Number	0	0	0	0	0	1	0	1
	Answer	**0**	**0**	**0**	**0**	**1**	**0**	**1**	**0**
Overflow **0**									

Left shift: 1

Correct? Y

Number 5 becomes 10

A right shift performs a: division.

Question 2:

Number	0	0	0	0	1	1	0	0
Answer	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**0**

0 Underflow

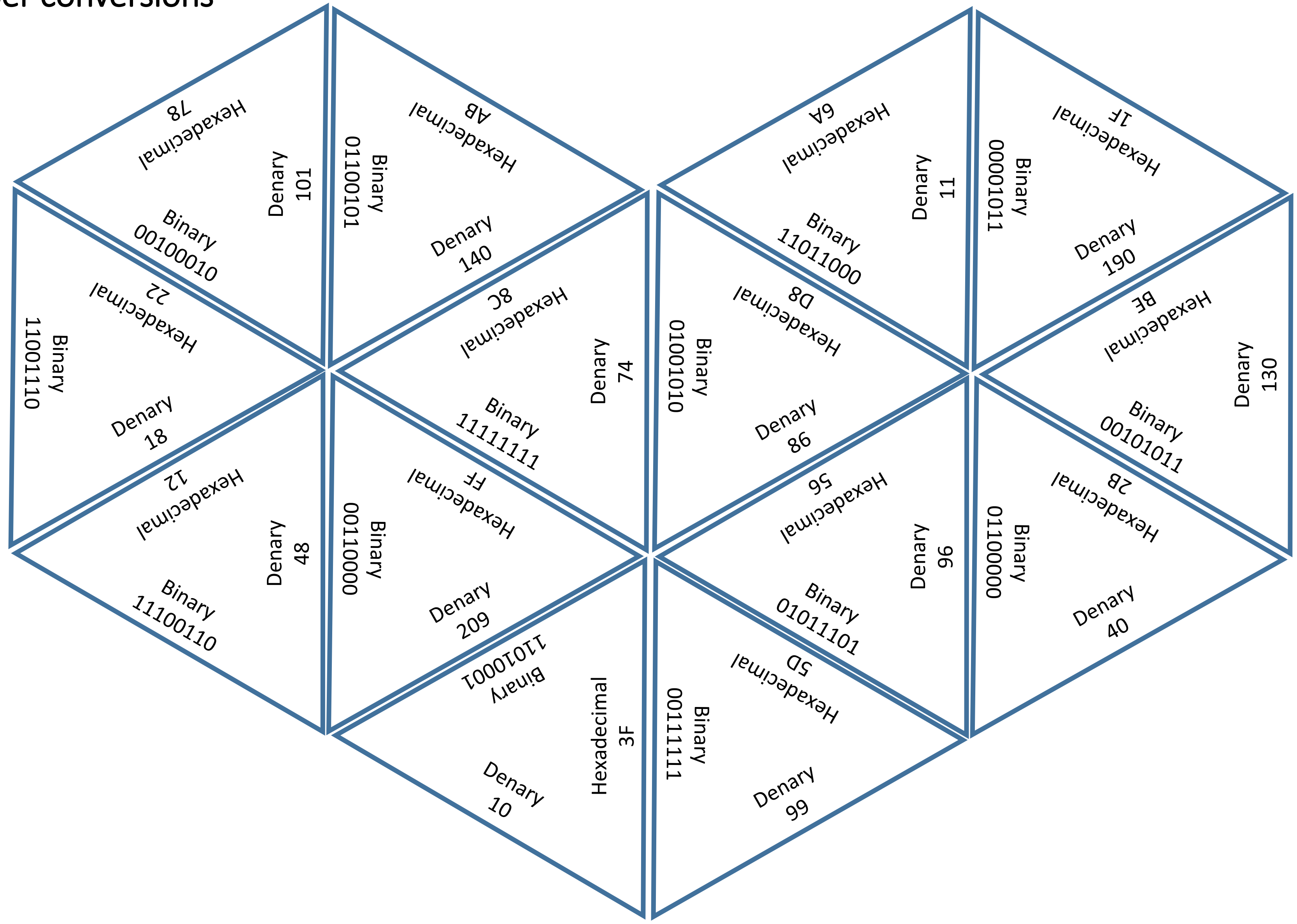
Right shift: 1

Correct? Y

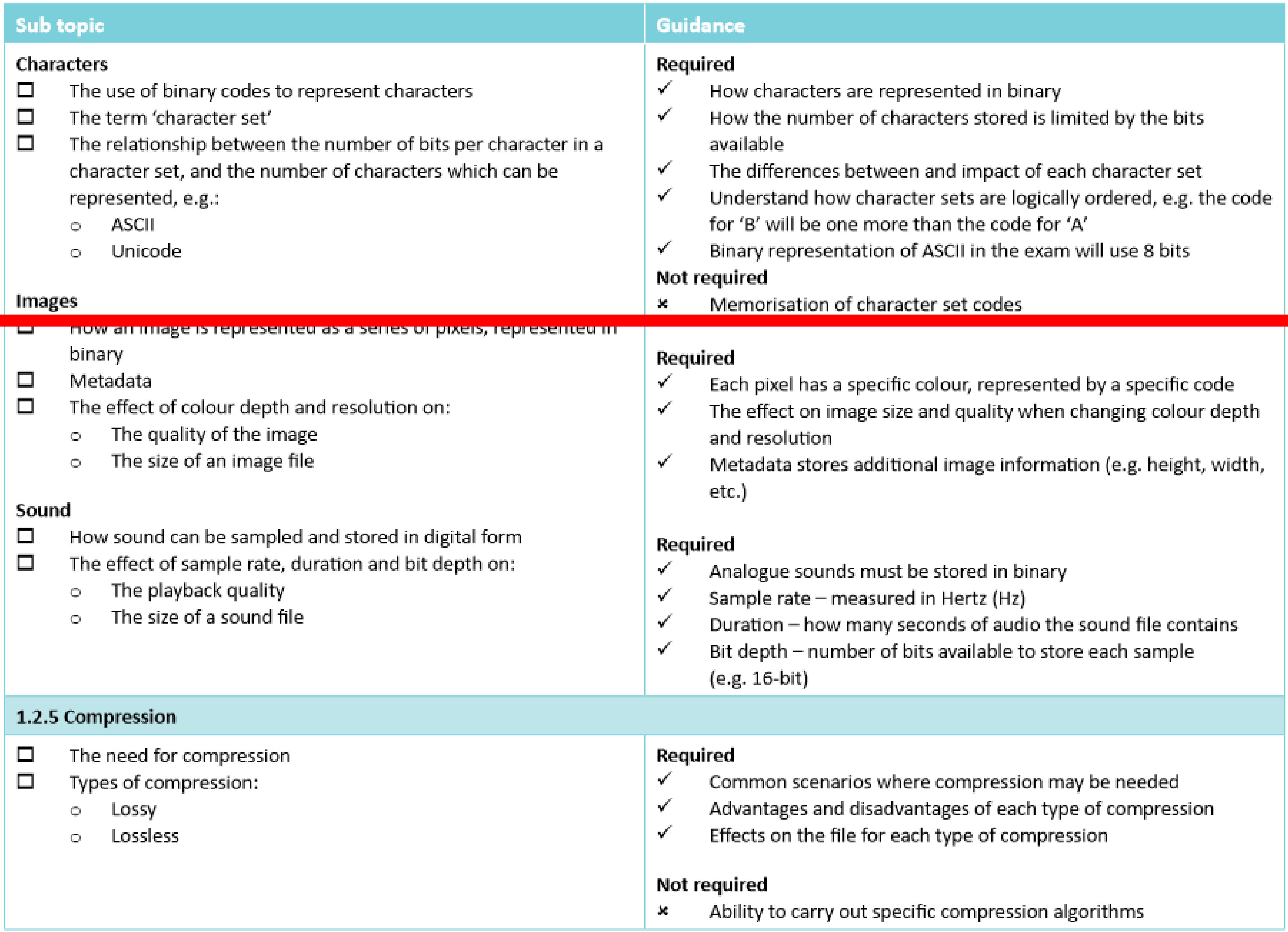
Number 12 becomes 6

J277 - 1.2 Memory & Storage

Number conversions



J277 - 1.2 Memory & Storage



Sub topic	Guidance
Characters ☐ The use of binary codes to represent characters ☐ The term 'character set' ☐ The relationship between the number of bits per character in a character set, and the number of characters which can be represented, e.g.: ○ ASCII ○ Unicode	**Required** ✓ How characters are represented in binary ✓ How the number of characters stored is limited by the bits available ✓ The differences between and impact of each character set ✓ Understand how character sets are logically ordered, e.g. the code for 'B' will be one more than the code for 'A' ✓ Binary representation of ASCII in the exam will use 8 bits **Not required** ✗ Memorisation of character set codes
Images ☐ How an image is represented as a series of pixels, represented in binary ☐ Metadata ☐ The effect of colour depth and resolution on: ○ The quality of the image ○ The size of an image file	**Required** ✓ Each pixel has a specific colour, represented by a specific code ✓ The effect on image size and quality when changing colour depth and resolution ✓ Metadata stores additional image information (e.g. height, width, etc.)
Sound ☐ How sound can be sampled and stored in digital form ☐ The effect of sample rate, duration and bit depth on: ○ The playback quality ○ The size of a sound file	**Required** ✓ Analogue sounds must be stored in binary ✓ Sample rate – measured in Hertz (Hz) ✓ Duration – how many seconds of audio the sound file contains ✓ Bit depth – number of bits available to store each sample (e.g. 16-bit)
1.2.5 Compression	
☐ The need for compression ☐ Types of compression: ○ Lossy ○ Lossless	**Required** ✓ Common scenarios where compression may be needed ✓ Advantages and disadvantages of each type of compression ✓ Effects on the file for each type of compression **Not required** ✗ Ability to carry out specific compression algorithms

J277 - 1.2 Memory & Storage

The use of binary codes to represent characters

Characters
These are uppercase and lowercase letters, the digits 0-9 and symbols like ?, + and £.

Character Sets
These are collections if characters that a computer recognises from their binary representation, used to convert characters to binary code and vice versa.

Button pressed on keyboard.

Binary signal sent to computer.

Computer translates code using character set.

Two important character sets

ASCII
- Each character is given a 7-bit binary code – so ASCII can represent 128 different characters.
- An extra bit (0) is added to the start of each binary code so each character uses 1 byte.
- The codes for numbers and letters are ordered (A comes before B before C…).

UNICODE
- Covers all major language, including ones that use different alphabets, like Greek, Russian and Chinese.
- Uses multiple bytes for each character.
- The first 128 characters in Unicode are the same as ASCII.

Character	Binary	Denary
A	0100 0001	65
B	0100 0010	66
C	0100 0011	67
a	0110 0001	97
b	0100 0010	98
c	0110 0011	99

J277 - 1.2 Memory & Storage

Text File Sizes

Use this formula to calculate file size:

File size (in bits)

= number of bits per character * number of characters

Example

How many bits would be needed to store “I’m a string, store me!” in 8 bit ASCII?

#Count the characters

I	‘	m		a		s	t	r	i	n	g	,		s	t	o	r	e		m	e	!

5 **10** **15** **20** **23**

#Use the formula

File size = 8 x 23 = ***184 bits***

How many bits would be needed to store “My phone number is 07584 587458!” in 8 bit ASCII? Write your answer in Bytes.

#Count the characters

32

#Use the formula

File size = 8 x 32 = ***32 bytes.***

J277 - 1.2 Memory & Storage

Sub topic	Guidance
Characters ☐ The use of binary codes to represent characters ☐ The term 'character set' ☐ The relationship between the number of bits per character in a character set, and the number of characters which can be represented, e.g.: ○ ASCII ○ Unicode	**Required** ✓ How characters are represented in binary ✓ How the number of characters stored is limited by the bits available ✓ The differences between and impact of each character set ✓ Understand how character sets are logically ordered, e.g. the code for 'B' will be one more than the code for 'A' ✓ Binary representation of ASCII in the exam will use 8 bits **Not required** ✗ Memorisation of character set codes
Images ☐ How an image is represented as a series of pixels, represented in binary ☐ Metadata ☐ The effect of colour depth and resolution on: ○ The quality of the image ○ The size of an image file	**Required** ✓ Each pixel has a specific colour, represented by a specific code ✓ The effect on image size and quality when changing colour depth and resolution ✓ Metadata stores additional image information (e.g. height, width, etc.)
Sound ☐ How sound can be sampled and stored in digital form ☐ The effect of sample rate, duration and bit depth on: ○ The playback quality ○ The size of a sound file	**Required** ✓ Analogue sounds must be stored in binary ✓ Sample rate – measured in Hertz (Hz) ✓ Duration – how many seconds of audio the sound file contains ✓ Bit depth – number of bits available to store each sample (e.g. 16-bit)
1.2.5 Compression	
☐ The need for compression ☐ Types of compression: ○ Lossy ○ Lossless	**Required** ✓ Common scenarios where compression may be needed ✓ Advantages and disadvantages of each type of compression ✓ Effects on the file for each type of compression **Not required** ✗ Ability to carry out specific compression algorithms

J277 - 1.2 Memory & Storage

The use of binary codes to represent characters

Bitmap
A type of image made up from lots of tiny dots, called **pixels.** The colour of each pixel is stored using binary code.

Image resolution
The number of pixels in a bitmap image. Often given as 'width x height.'

Metadata

Data stored in a file which contains information ab out the file. Helps the computer to recreate the image on screen from the binary data in each pixel.

Examples of Metadata include:

- Height and Width
- Colour depth
- Resolution
- File format
- Data created
- Date last edited

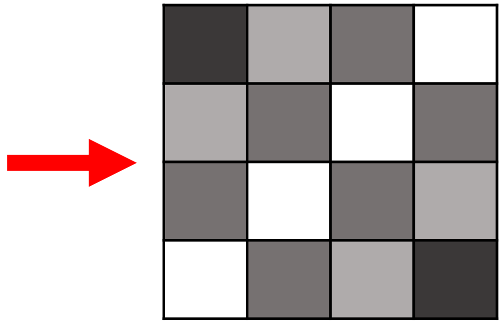
Colour Depth
The number of bits used to represent each pixel. The number of colours that can be used for a given colour depth follows this formula:

$$2^{n} \quad \text{(where n = colour depth)}$$

Example
Colour depth = 2 bits

- Number of colours = 2^2 = 4
- In this example: 00 = white, 01 = light grey, 10 dark grey & 11 = black.

11	01	10	00
01	10	00	10
10	00	10	01
00	10	01	11



J277 - 1.2 Memory & Storage

Image File Sizes

Use this formula to calculate file size:

File size (in bits)

= image resolution x colour depth
= width x height x colour depth

Increasing the image resolution or colour depth will usually give a high quality image, but larger file size.

Example

Calculate the size in kB of a 100x100 pixel image with a colour depth of 16 bits.

#Calculating the size
File size = 100x100x16
= 160,000 bits

#Calculating in KB
160,000 bits
= 160,000 / 8 = 20,000 bytes
= 20, 000 / 1000 = 20 KB

Example

Calculate the size in MB of a 800 pixels (width) × 600 pixels (height) with a colour depth of 24 bits.

#Calculating the size
File size = 800x600x24
= 11,520,000 bits

#Calculating in MB
11,520,000 bits
= 11,520,000 bits / 8 = 1,440,000 bytes
= 1,440,000 / 1000 = 1,440 KB
1,400KB / 1000 = 1.44MB

Example

Calculate the size in MB of a 750 pixels (width) × 900 pixels (height) with a colour depth of 32 bits.

#Calculating the size
File size = 750x900x32
= 21,600,000 bits

#Calculating in GB
21,600,000 bits
= 21,600,000 bits / 8 = 2,700,000 bytes
= 2,700,000 / 1000 = 2,700 KB
2,700 KB / 1000 = 27MB

J277 - 1.2 Memory & Storage

Sub topic	Guidance
Characters ☐ The use of binary codes to represent characters ☐ The term 'character set' ☐ The relationship between the number of bits per character in a character set, and the number of characters which can be represented, e.g.: ○ ASCII ○ Unicode	**Required** ✓ How characters are represented in binary ✓ How the number of characters stored is limited by the bits available ✓ The differences between and impact of each character set ✓ Understand how character sets are logically ordered, e.g. the code for 'B' will be one more than the code for 'A' ✓ Binary representation of ASCII in the exam will use 8 bits **Not required** ✗ Memorisation of character set codes
Images ☐ How an image is represented as a series of pixels, represented in binary ☐ Metadata ☐ The effect of colour depth and resolution on: ○ The quality of the image ○ The size of an image file	**Required** ✓ Each pixel has a specific colour, represented by a specific code ✓ The effect on image size and quality when changing colour depth and resolution ✓ Metadata stores additional image information (e.g. height, width, etc.)
Sound ☐ How sound can be sampled and stored in digital form ☐ The effect of sample rate, duration and bit depth on: ○ The playback quality ○ The size of a sound file	**Required** ✓ Analogue sounds must be stored in binary ✓ Sample rate – measured in Hertz (Hz) ✓ Duration – how many seconds of audio the sound file contains ✓ Bit depth – number of bits available to store each sample (e.g. 16-bit)
1.2.5 Compression	
☐ The need for compression ☐ Types of compression: ○ Lossy ○ Lossless	**Required** ✓ Common scenarios where compression may be needed ✓ Advantages and disadvantages of each type of compression ✓ Effects on the file for each type of compression **Not required** ✗ Ability to carry out specific compression algorithms

J277 - 1.2 Memory & Storage

Sampling

Key definitions

We hear sound as 'analogue' – tones and rhythms blend together smoothly.

If sound has to be handled by the computer – needs to be converted to binary.

'Digitalise' using an input device.

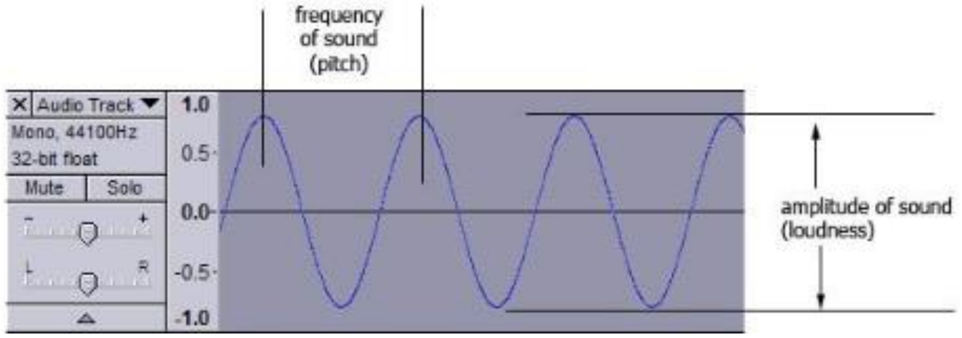
Sampling – Converting analogue sound wave into digital data that can be read and stored by a computer.

Sample frequency – The number of audio samples captured every second. Measured in Hertz (Hz)

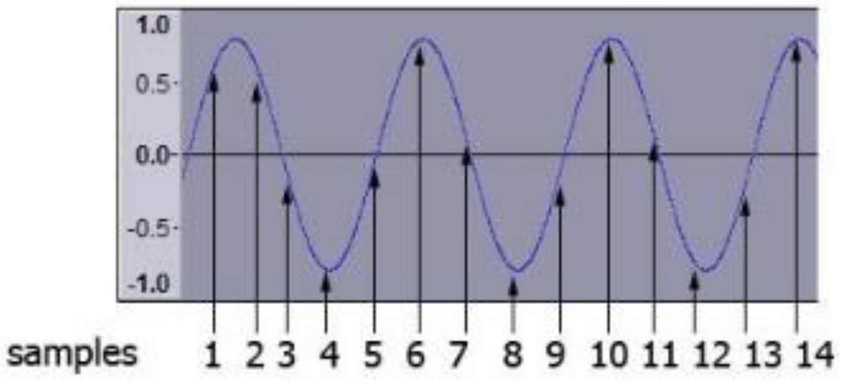
Bit depth – Number of bits available for each sample.

Sound Sampling process

Typical sound wave



The height of the wave is measured (sampled) at regular intervals.
This is then turned into binary and stored.



J277 - 1.2 Memory & Storage

Sound File Sizes

Use this formula to calculate file size:

File size (in bits)

= Sample rate (in Hz) x bit depth x length (in Secs)

A higher sample rate or bit depth will give a higher quality sound file, but will increase the file size.

Example

Calculate the file size in MB of a 50 second audio recording with a sample rate of 40kHz and a bit depth of 8 bits.

#Calculating the size
File size = 40kHz = 40,000 Hz
=50x40,000x8
= 16,000,000 bits

#Calculating in MB
16,000,000 bits
= 16,000,000 / 8 = 2,000,000 bytes
= 2,000,000 / 1000 = 2000 KB
= 2000 KB / 1000 = 2GB

Example

Calculate the file size in MB for an audio quality recording with a 16-bit depth, sample rate of 44,100 Hz (samples per second) and duration of 3 minutes.

#Calculating the size
File size = 16x44,100x180
= 127,008,000 bits

#Calculating in MB
127,008,000 bits
= 127,008,000 / 8 = 15,876,000 bytes
= 15,876,000 / 1000 = 15,876 KB
15,876 / 1000 = 15.876 MB

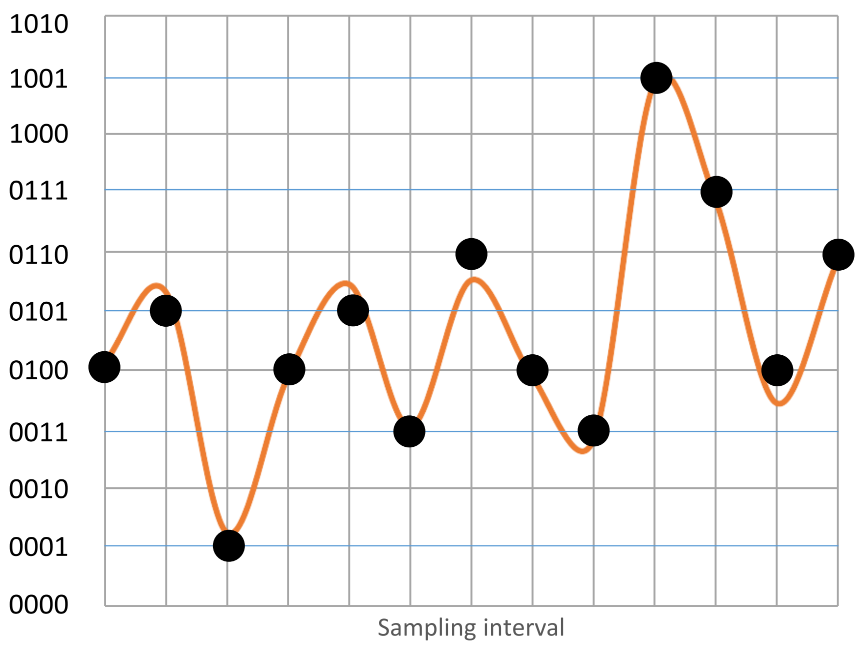
Example

Calculate the file size in MB for an audio quality recording with a 24-bit depth, sample rate of 48kHz (samples per second) and duration of 5 minutes.

#Calculating the size
File size = 48kHz = 48,000 Hz
= 5 mins = 300 secs
24x48,000x300
= 345,600,000 bits
#Calculating in MB
= 345,600,000 / 8 = 43,200,000 bytes
= 43,200,000 / 1000 = 43,200 KB
43,200KB / 1000 = 43.2 MB

J277 - 1.2 Memory & Storage

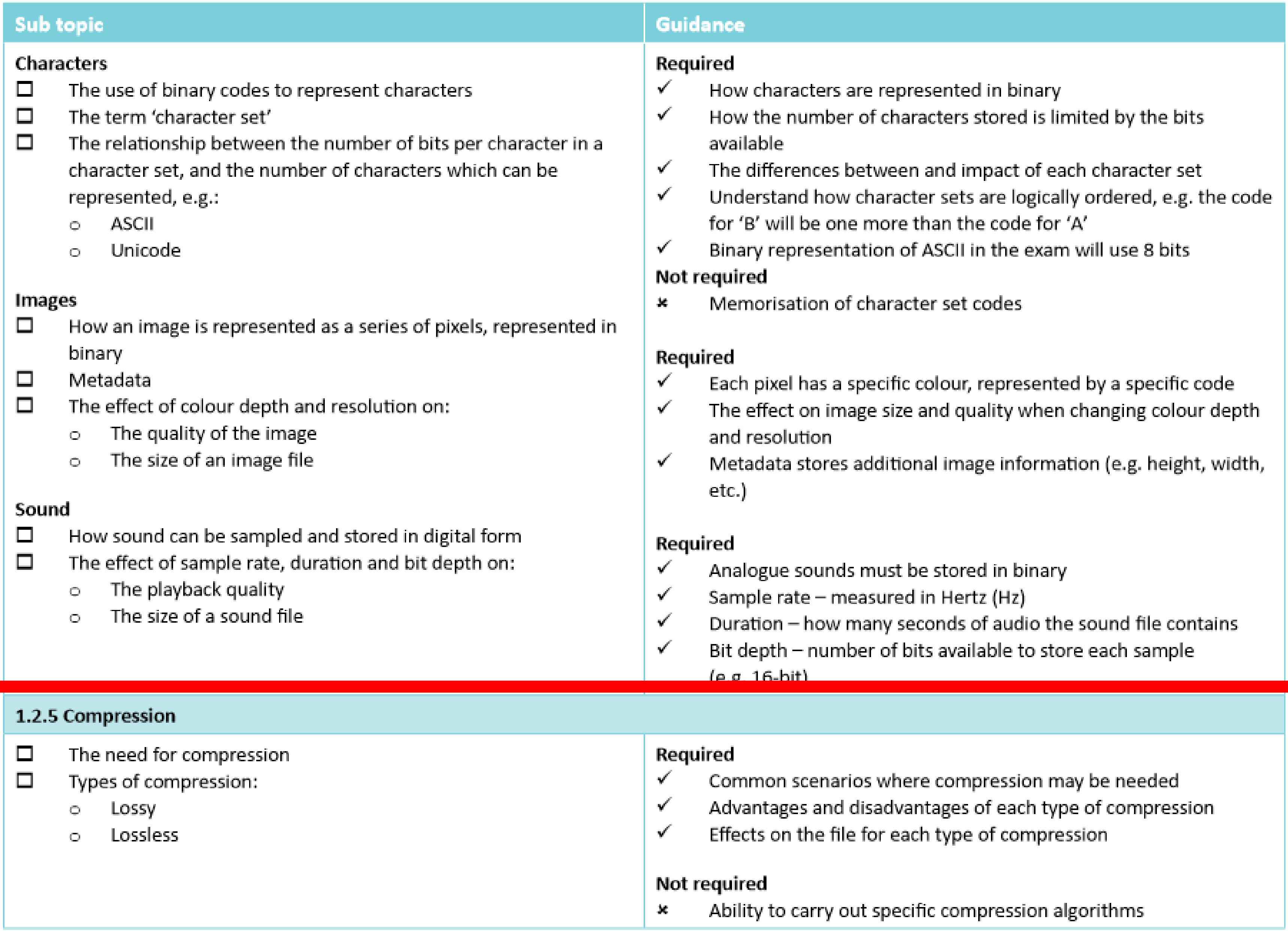
The effect of doubling the bit rate on the quality of the sound and file size



Binary for the sound could be: 0100 0101 0001 0100 0101 0011 0110 0100 0011 1001 0111 0100 0110 | 52 | bits.

The quality has: Increased because the digital value is closer to the original wave.

J277 - 1.2 Memory & Storage



Sub topic	Guidance
Characters ☐ The use of binary codes to represent characters ☐ The term 'character set' ☐ The relationship between the number of bits per character in a character set, and the number of characters which can be represented, e.g.: ○ ASCII ○ Unicode	**Required** ✓ How characters are represented in binary ✓ How the number of characters stored is limited by the bits available ✓ The differences between and impact of each character set ✓ Understand how character sets are logically ordered, e.g. the code for 'B' will be one more than the code for 'A' ✓ Binary representation of ASCII in the exam will use 8 bits **Not required** ✗ Memorisation of character set codes
Images ☐ How an image is represented as a series of pixels, represented in binary ☐ Metadata ☐ The effect of colour depth and resolution on: ○ The quality of the image ○ The size of an image file	**Required** ✓ Each pixel has a specific colour, represented by a specific code ✓ The effect on image size and quality when changing colour depth and resolution ✓ Metadata stores additional image information (e.g. height, width, etc.)
Sound ☐ How sound can be sampled and stored in digital form ☐ The effect of sample rate, duration and bit depth on: ○ The playback quality ○ The size of a sound file	**Required** ✓ Analogue sounds must be stored in binary ✓ Sample rate – measured in Hertz (Hz) ✓ Duration – how many seconds of audio the sound file contains ✓ Bit depth – number of bits available to store each sample (e.g. 16-bit)
1.2.5 Compression	
☐ The need for compression ☐ Types of compression: ○ Lossy ○ Lossless	**Required** ✓ Common scenarios where compression may be needed ✓ Advantages and disadvantages of each type of compression ✓ Effects on the file for each type of compression **Not required** ✗ Ability to carry out specific compression algorithms

J277 - 1.2 Memory & Storage

Data Compression

What is Data Compression?

Making file sizes smaller, whilst trying to stay as true to the original as possible.

*Less **storage** space used.*

*Streaming/Downloading compressed files takes less **bandwidth**.*

Some services (eg. Email) have file size limits, compression can get below limits.

	Pros	Cons
<u>Lossy</u> Permanently removes data from the file.	• Small file sizes. • Ideal for web use due to file size. • Commonly used – Lots of software and read lossy files.	• Looses data – Can't be reversed to the original. • Quality degrades. • Can't be used on text / software.
<u>Lossless</u> Temporarily removes data to store the file and restores it to the original state when opened.	• No loss in quality. • File can be turned back to the original. • Can be used of text / software.	• Larger than lossy files.

J277 - 1.2 Memory & Storage

Units of data storage

Case study:

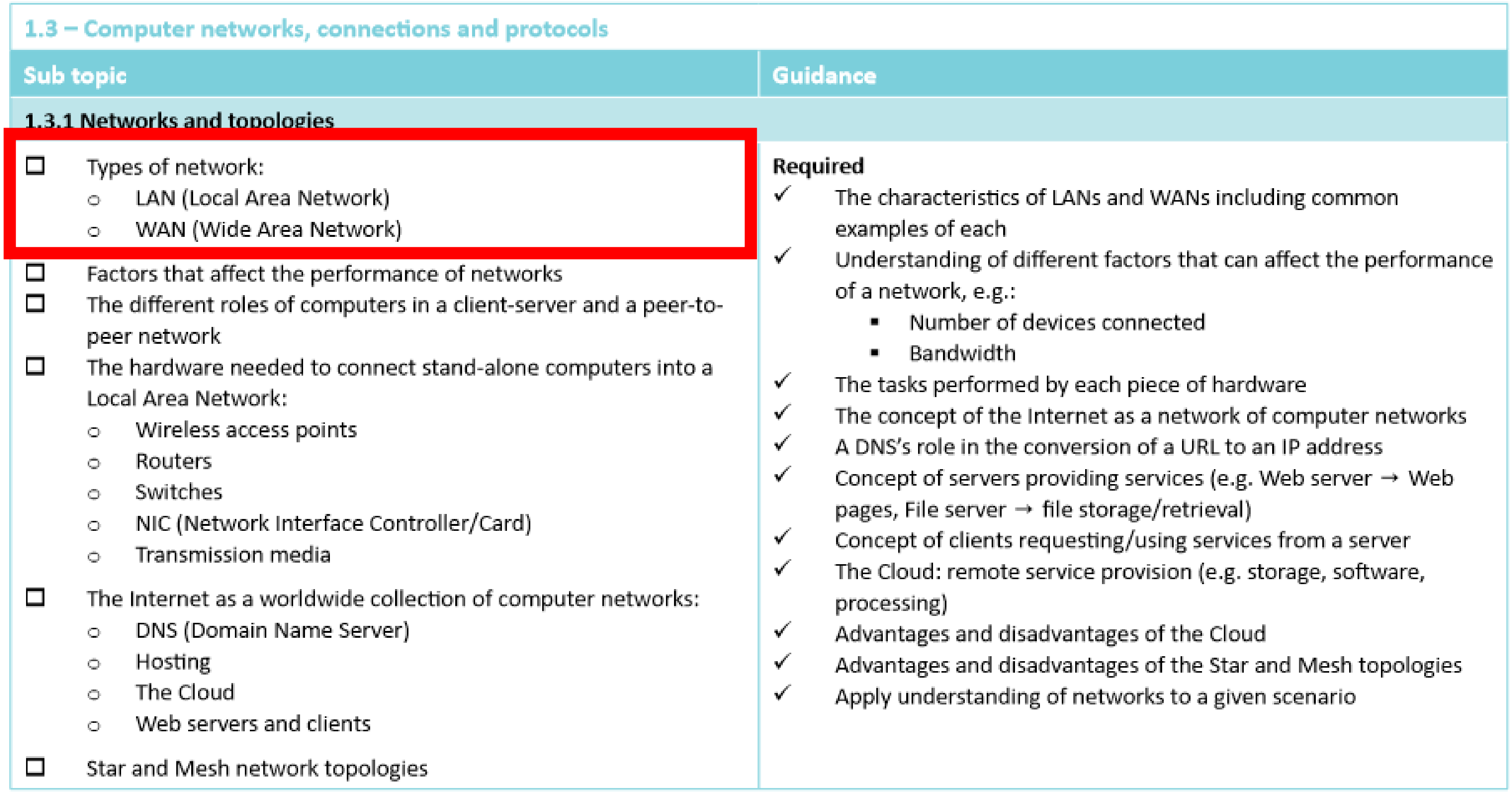
Elizabeth is looking to upgrade her phone. She is choosing between model A and model B.

	Model A	Model B
Storage capacity	64GB	256GB
Camera	8 mega-pixel, 24bit colour	12 mega-pixel, 24 bit colour
Video	1080p (2 mega-pixel) at 24 fps, 30 fps, or 60 fps	4K HD (8 mega-pixel) at 24 fps, 30 fps, or 60 fps

Consider that 32GB of data storage will be used for apps. Assume 50% compression.

Maximum number of photographs that could be stored on the phone:	8,000,000 bit resolution. x 24 bit = 192,000,000 bits. Divide by 8 for bytes = 24,000,000 bytes or 24MB Per photograph. 32,000 (32GB) divided by 24 = 1,333 uncompressed photos. x 2 due to 50% compression = 2,666 photos.	12,000,000 bit resolution. X 24 bit = 288.000.000 bits. Divide by 8 for bytes = 36,000,000 bytes or 36MB Per photograph. 224,000 (224GB) divided by 36 = 6,222 uncompressed photos. x 2 due to 50% compression = 12,444 photos.

J277 - 1.3 Computer networks, connections and protocols



1.3 – Computer networks, connections and protocols

Sub topic	Guidance
1.3.1 Networks and topologies	
☐ Types of network: ○ LAN (Local Area Network) ○ WAN (Wide Area Network) ☐ Factors that affect the performance of networks ☐ The different roles of computers in a client-server and a peer-to-peer network ☐ The hardware needed to connect stand-alone computers into a Local Area Network: ○ Wireless access points ○ Routers ○ Switches ○ NIC (Network Interface Controller/Card) ○ Transmission media ☐ The Internet as a worldwide collection of computer networks: ○ DNS (Domain Name Server) ○ Hosting ○ The Cloud ○ Web servers and clients ☐ Star and Mesh network topologies	**Required** ✓ The characteristics of LANs and WANs including common examples of each ✓ Understanding of different factors that can affect the performance of a network, e.g.: ▪ Number of devices connected ▪ Bandwidth ✓ The tasks performed by each piece of hardware ✓ The concept of the Internet as a network of computer networks ✓ A DNS's role in the conversion of a URL to an IP address ✓ Concept of servers providing services (e.g. Web server → Web pages, File server → file storage/retrieval) ✓ Concept of clients requesting/using services from a server ✓ The Cloud: remote service provision (e.g. storage, software, processing) ✓ Advantages and disadvantages of the Cloud ✓ Advantages and disadvantages of the Star and Mesh topologies ✓ Apply understanding of networks to a given scenario

J277 - 1.3 Computer networks, connections and protocols

What is a Network?

A computer network is where two, or more, computing devices are connected together so that they can share resources.
Computers that are not connected to a network are called stand-alone computers.

Advantages of Networks

- Sharing devices such as printers saves **money**.
- Site (software) licences are likely to be **cheaper** than buying several standalone licences.
- Files can easily be **shared** between users.
- Network users can **communicate** by email and instant messenger.
- **Security** is **good** - users cannot see other users' files unlike on stand-alone machines.
- Data is easy to **backup** as all the data is stored on the file server.

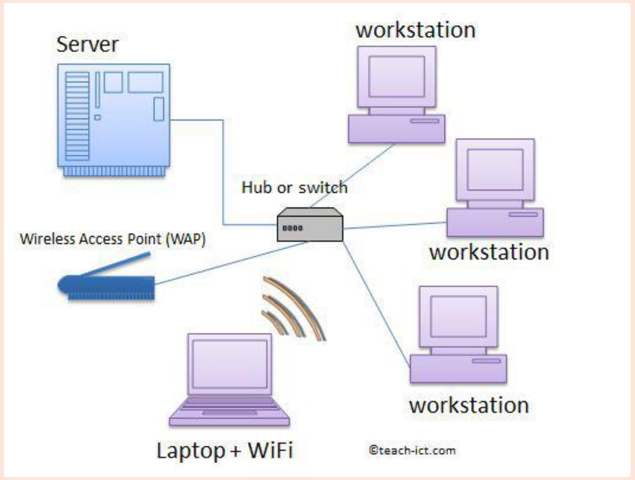
Disadvantages of Networks

- Purchasing the network cabling and file servers can be **expensive**.
- Managing a large network is complicated, **requires training** and a network manager usually needs to be employed.
- If the file server breaks down the files on the file server become **inaccessible**.
- **Viruses** can spread to other computers throughout a computer network.
- There is a danger of **hacking**, particularly with wide area networks.

J277 - 1.3 Computer networks, connections and protocols

Types of networks

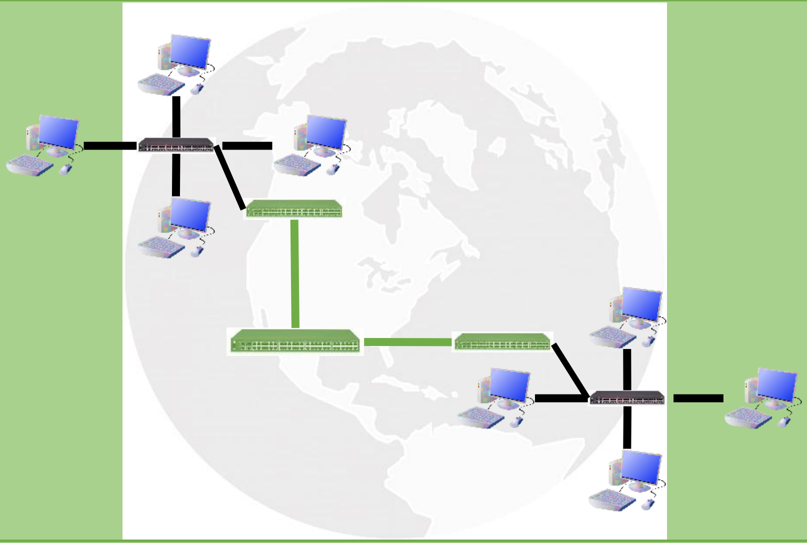
Representation of a local area network (LAN):



Description of a local area network:

- Covers a small geographic area located on a single site.
- All the hardware for the LAN is owned by the organisation using it.
- LANs are wired or wireless.

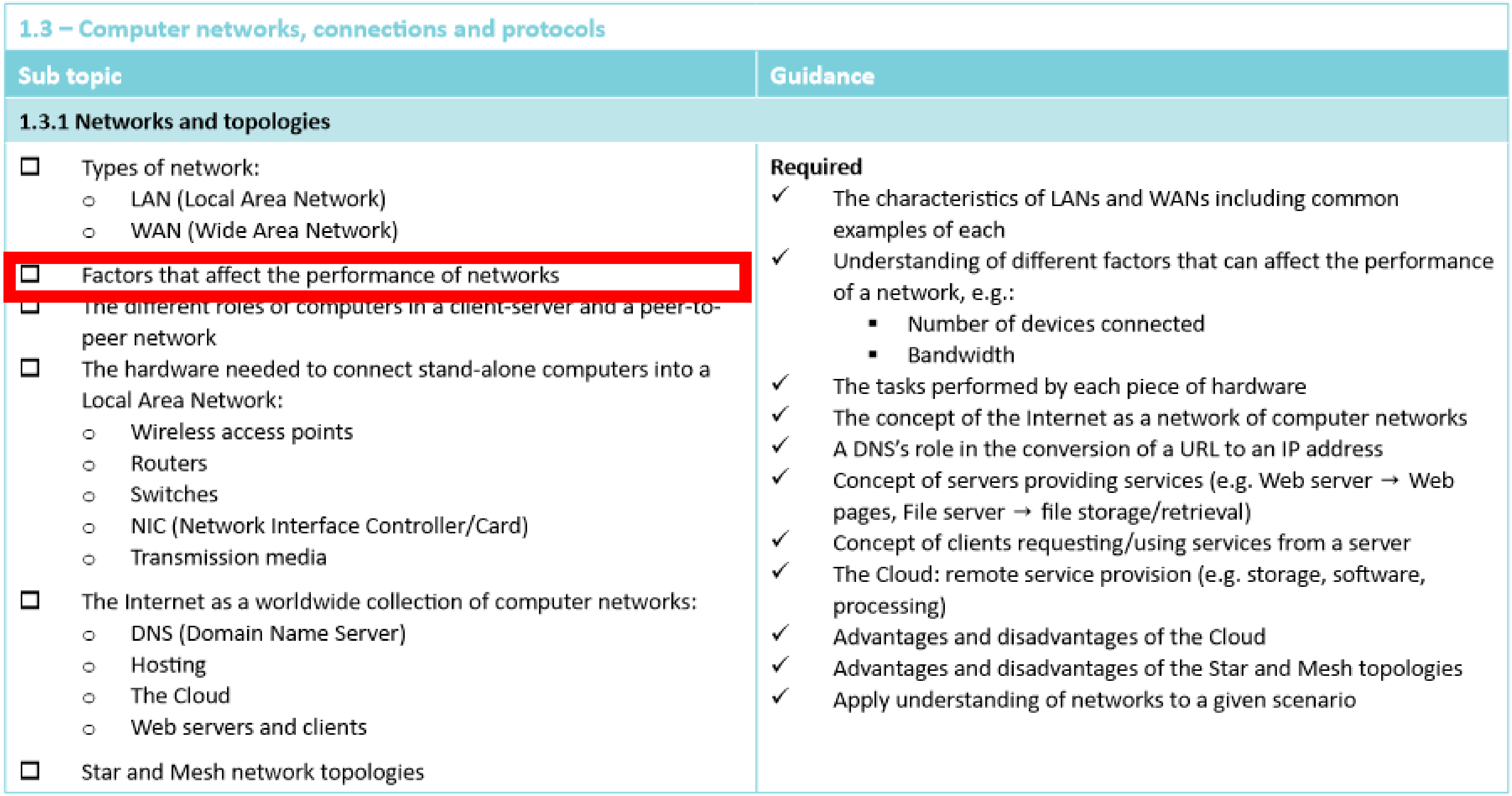
Representation of a wide area network (WAN):



Description of a wide area network:

- Covers a large geographic area that connects LANs together.
- Infrastructure between the LANs is hired from telecommunication companies who own and manage it.
- WANs are connected with telephone lines, fibre optic cables or satellite links.

J277 - 1.3 Computer networks, connections and protocols



1.3 – Computer networks, connections and protocols

Sub topic	Guidance
1.3.1 Networks and topologies	
☐ Types of network: ○ LAN (Local Area Network) ○ WAN (Wide Area Network) ☐ Factors that affect the performance of networks ☐ The different roles of computers in a client-server and a peer-to-peer network ☐ The hardware needed to connect stand-alone computers into a Local Area Network: ○ Wireless access points ○ Routers ○ Switches ○ NIC (Network Interface Controller/Card) ○ Transmission media ☐ The Internet as a worldwide collection of computer networks: ○ DNS (Domain Name Server) ○ Hosting ○ The Cloud ○ Web servers and clients ☐ Star and Mesh network topologies	**Required** ✓ The characteristics of LANs and WANs including common examples of each ✓ Understanding of different factors that can affect the performance of a network, e.g.: ▪ Number of devices connected ▪ Bandwidth ✓ The tasks performed by each piece of hardware ✓ The concept of the Internet as a network of computer networks ✓ A DNS's role in the conversion of a URL to an IP address ✓ Concept of servers providing services (e.g. Web server → Web pages, File server → file storage/retrieval) ✓ Concept of clients requesting/using services from a server ✓ The Cloud: remote service provision (e.g. storage, software, processing) ✓ Advantages and disadvantages of the Cloud ✓ Advantages and disadvantages of the Star and Mesh topologies ✓ Apply understanding of networks to a given scenario

J277 - 1.3 Computer networks, connections and protocols

Some of the factors which affect network performance are:

The bandwidth of the network

Bandwidth refers to the maximum amount of data that can be transmitted over a network in a given amount of time.

Higher bandwidth allows for faster data transfer rates, resulting in improved network performance.

The number of users on the network

Too many users or devices on the same network can cause the network to slow down if there is insufficient bandwidth for the data.

The error rate

Less reliable connections increase the number of errors that occur when data is being transferred, resulting in data having to be resent.

The transmission media used

Wired connections give better performance than wireless connections.

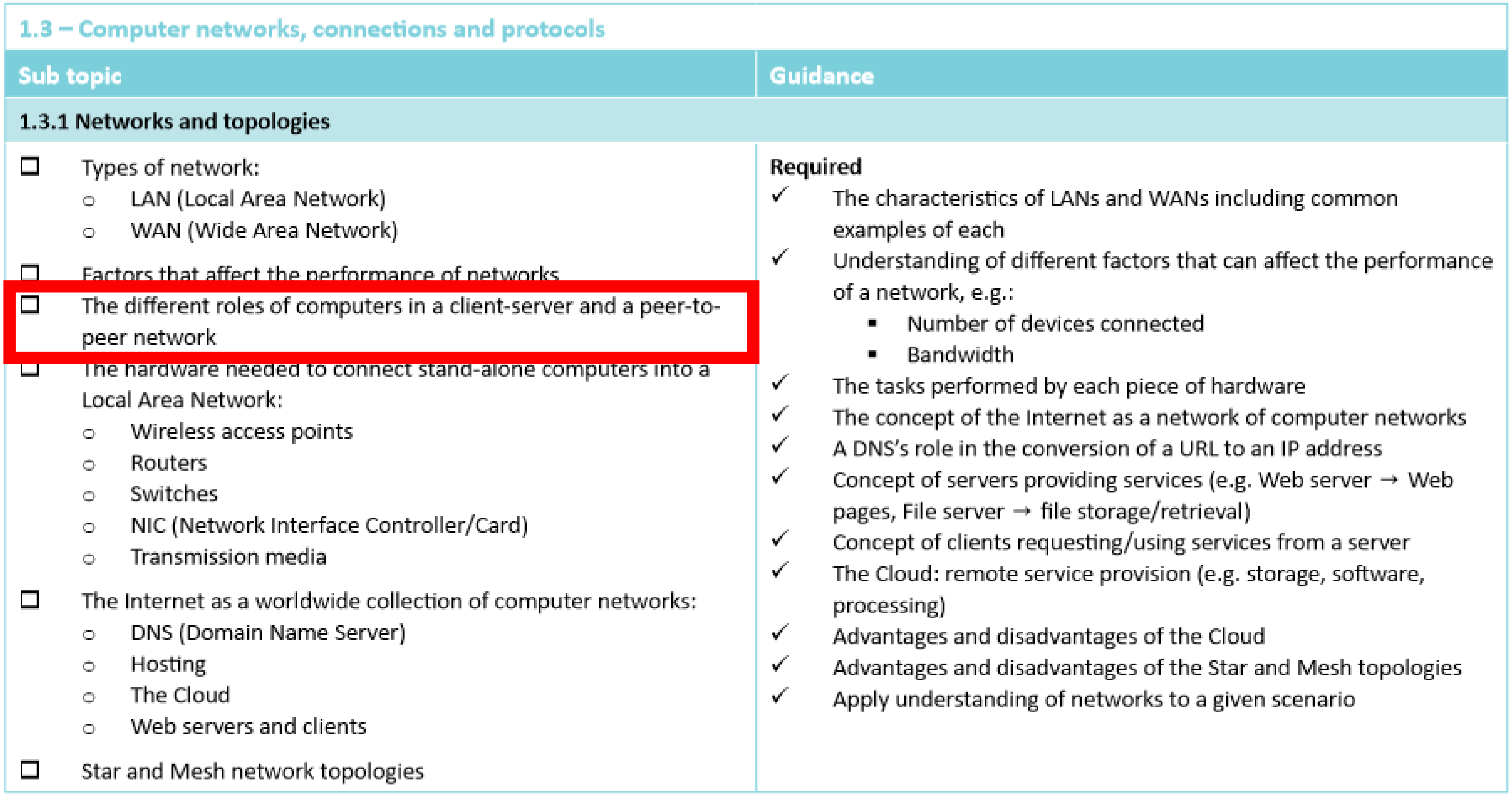
Fibre optic cables have a higher bandwidth than copper cables.

Latency

The delay from transmitting data to receiving it.

Lower latency is desirable as it reduces the delay in data transmission, resulting in quicker response times.

J277 - 1.3 Computer networks, connections and protocols



1.3 – Computer networks, connections and protocols

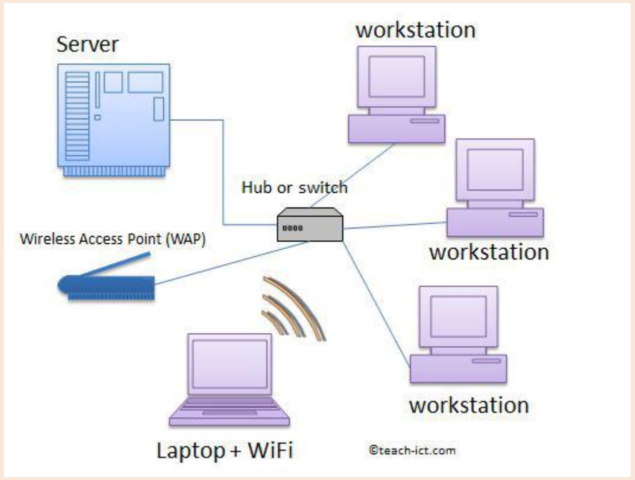
Sub topic	Guidance
1.3.1 Networks and topologies	
☐ Types of network: ○ LAN (Local Area Network) ○ WAN (Wide Area Network) ☐ Factors that affect the performance of networks ☐ The different roles of computers in a client-server and a peer-to-peer network ☐ The hardware needed to connect stand-alone computers into a Local Area Network: ○ Wireless access points ○ Routers ○ Switches ○ NIC (Network Interface Controller/Card) ○ Transmission media ☐ The Internet as a worldwide collection of computer networks: ○ DNS (Domain Name Server) ○ Hosting ○ The Cloud ○ Web servers and clients ☐ Star and Mesh network topologies	**Required** ✓ The characteristics of LANs and WANs including common examples of each ✓ Understanding of different factors that can affect the performance of a network, e.g.: ▪ Number of devices connected ▪ Bandwidth ✓ The tasks performed by each piece of hardware ✓ The concept of the Internet as a network of computer networks ✓ A DNS's role in the conversion of a URL to an IP address ✓ Concept of servers providing services (e.g. Web server → Web pages, File server → file storage/retrieval) ✓ Concept of clients requesting/using services from a server ✓ The Cloud: remote service provision (e.g. storage, software, processing) ✓ Advantages and disadvantages of the Cloud ✓ Advantages and disadvantages of the Star and Mesh topologies ✓ Apply understanding of networks to a given scenario

J277 - 1.3 Computer networks, connections and protocols

Network Topologies

Both local area networks and wide area networks can operate as either a client-server or a peer-to-peer approach when sharing data.

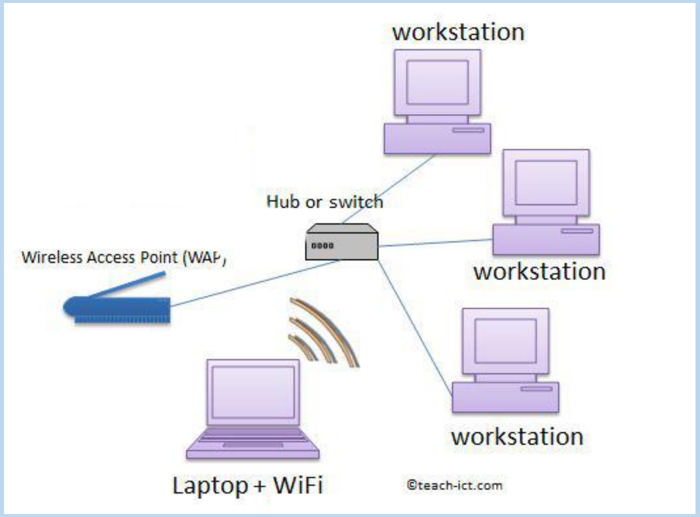
Client-server model



Description of a client-server network

- A server controls access and security to one shared file store.
- A server manages access to the internet.
- A server manages printing jobs.
- A server provides email services.
- A server runs a backup of data.
- A client makes requests to the server for data and connections.

Peer-to-peer model



- A peer is equal to all other peers, they serve their own files to each other.
- Each peer is responsible for its own security.
- Each peer is responsible for its own backup.
- Peers usually have their own printers. You can send print jobs to another peer to process, but that peer would need to be switched on to be able to communicate with the connected printer.

J277 - 1.3 Computer networks, connections and protocols

Advantages and disadvantages of client-server and peer-to-peer networks.

Client-server model

Advantages:

- Resource sharing.
- Easier to take backups of all shared data.
- Easier to install software updates.
- Security – Passwords & access levels.

Disadvantages:

- Expensive to set up and maintain.
- Requires IT specialists to maintain.
- A single point of failure. Users will lose access if the server fails.
- Requires IT specialists to maintain.

Peer-to-peer model

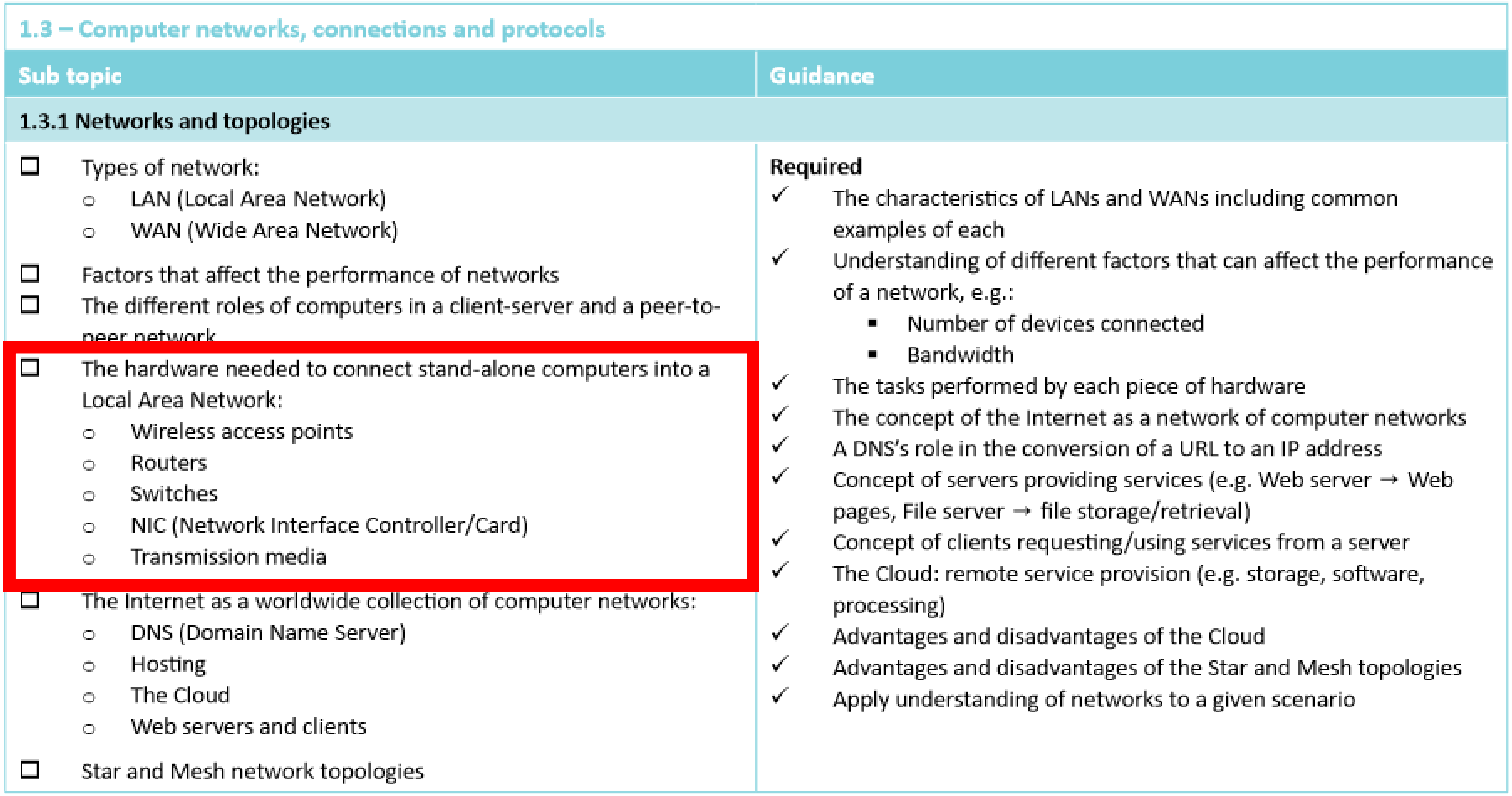
Advantages:

- Very easy to maintain without expertise employed.
- No dependency on a single server.
- Cheaper to set up. No expensive hardware required.

Disadvantages:

- The network is less secure.
- Users will need to manage their own backups.
- Can be complicated to update.

J277 - 1.3 Computer networks, connections and protocols

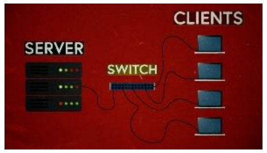


1.3 – Computer networks, connections and protocols

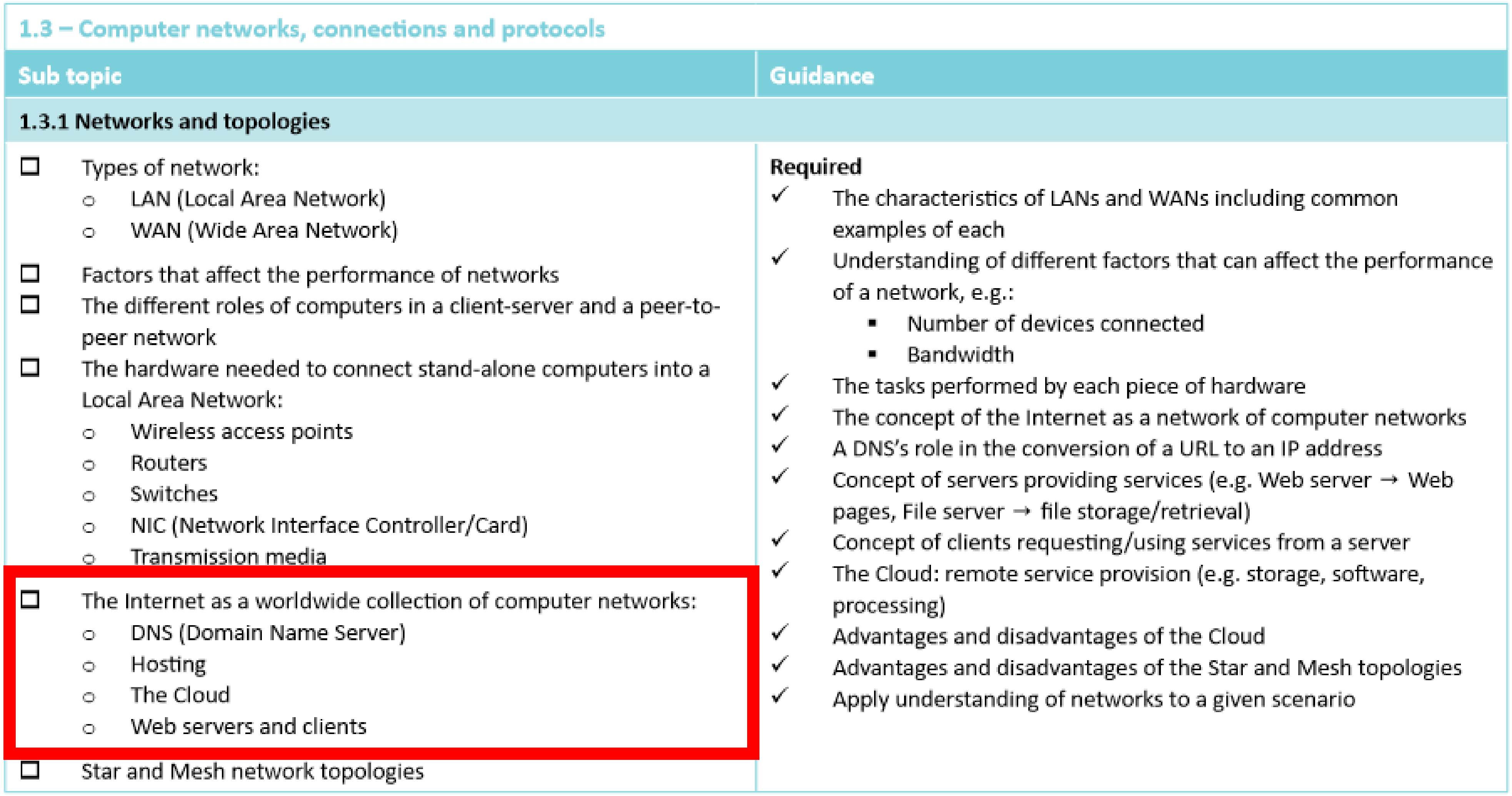
Sub topic	Guidance
1.3.1 Networks and topologies	
☐ Types of network: ○ LAN (Local Area Network) ○ WAN (Wide Area Network) ☐ Factors that affect the performance of networks ☐ The different roles of computers in a client-server and a peer-to-peer network ☐ The hardware needed to connect stand-alone computers into a Local Area Network: ○ Wireless access points ○ Routers ○ Switches ○ NIC (Network Interface Controller/Card) ○ Transmission media ☐ The Internet as a worldwide collection of computer networks: ○ DNS (Domain Name Server) ○ Hosting ○ The Cloud ○ Web servers and clients ☐ Star and Mesh network topologies	**Required** ✓ The characteristics of LANs and WANs including common examples of each ✓ Understanding of different factors that can affect the performance of a network, e.g.: ▪ Number of devices connected ▪ Bandwidth ✓ The tasks performed by each piece of hardware ✓ The concept of the Internet as a network of computer networks ✓ A DNS's role in the conversion of a URL to an IP address ✓ Concept of servers providing services (e.g. Web server → Web pages, File server → file storage/retrieval) ✓ Concept of clients requesting/using services from a server ✓ The Cloud: remote service provision (e.g. storage, software, processing) ✓ Advantages and disadvantages of the Cloud ✓ Advantages and disadvantages of the Star and Mesh topologies ✓ Apply understanding of networks to a given scenario

J277 - 1.3 Computer networks, connections and protocols

Network Hardware

Hardware	Function
Network Interface Controller (NIC)	A network interface controller (NIC) provides a method of connecting to a network.
Switch 	Switches receive and transmits data between devices on a LAN using Mac addresses.
Transmission Media	Wired or wireless radio waves that carry data signals from one devices to another.
Wireless Access Point (WAP)	Allows devices to connect to a network wirelessly. Similar to a switch.
Router	Transmits data between networks by directing data as 'packets' to their destination.

J277 - 1.3 Computer networks, connections and protocols



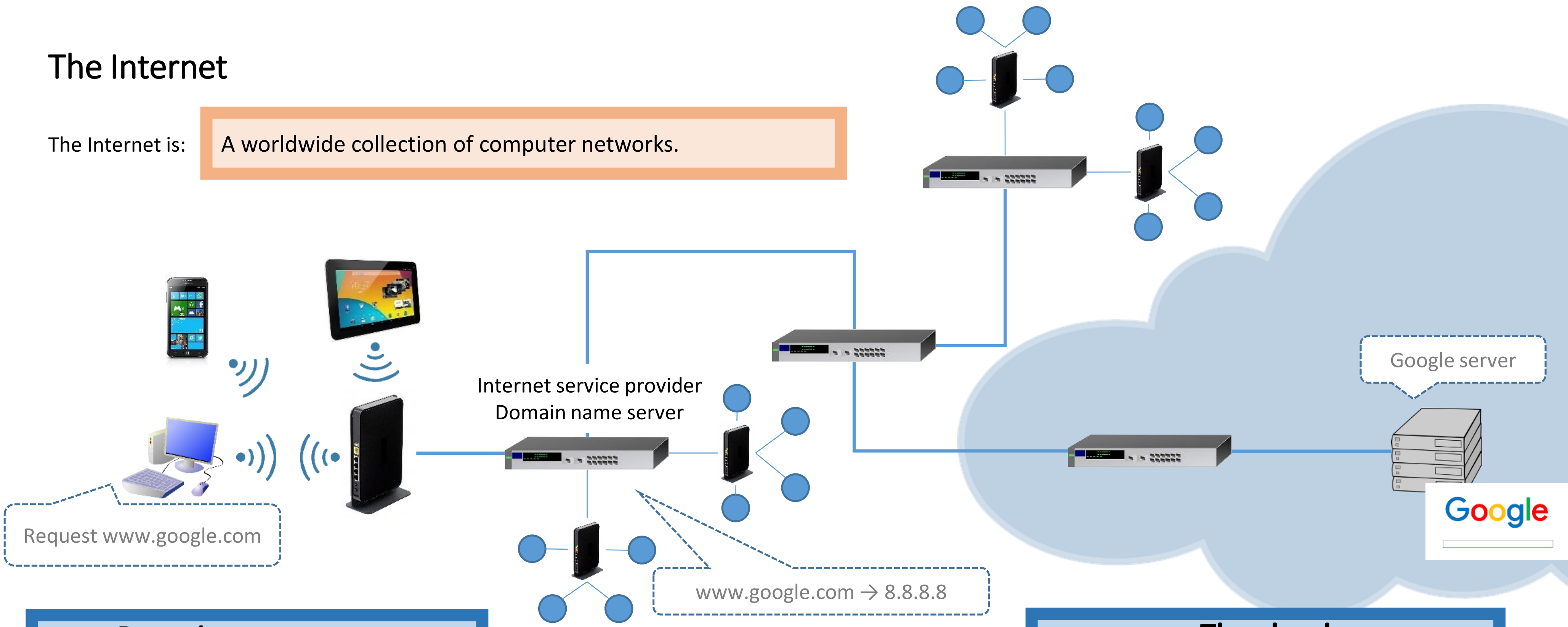
1.3 – Computer networks, connections and protocols

Sub topic	Guidance
1.3.1 Networks and topologies	
☐ Types of network: ○ LAN (Local Area Network) ○ WAN (Wide Area Network) ☐ Factors that affect the performance of networks ☐ The different roles of computers in a client-server and a peer-to-peer network ☐ The hardware needed to connect stand-alone computers into a Local Area Network: ○ Wireless access points ○ Routers ○ Switches ○ NIC (Network Interface Controller/Card) ○ Transmission media ☐ The Internet as a worldwide collection of computer networks: ○ DNS (Domain Name Server) ○ Hosting ○ The Cloud ○ Web servers and clients ☐ Star and Mesh network topologies	**Required** ✓ The characteristics of LANs and WANs including common examples of each ✓ Understanding of different factors that can affect the performance of a network, e.g.: ▪ Number of devices connected ▪ Bandwidth ✓ The tasks performed by each piece of hardware ✓ The concept of the Internet as a network of computer networks ✓ A DNS's role in the conversion of a URL to an IP address ✓ Concept of servers providing services (e.g. Web server → Web pages, File server → file storage/retrieval) ✓ Concept of clients requesting/using services from a server ✓ The Cloud: remote service provision (e.g. storage, software, processing) ✓ Advantages and disadvantages of the Cloud ✓ Advantages and disadvantages of the Star and Mesh topologies ✓ Apply understanding of networks to a given scenario

J277 - 1.3 Computer networks, connections and protocols

The Internet

The Internet is: A worldwide collection of computer networks.



Domain name server

A service made of many domain name servers that store domain names and matching IP addresses.

1. Browser sends URL to DNS.
2. DNS finds matching IP address and sends it back to the browser.
3. Browser requests web page from the web server at the IP address.
4. Web server processes the request and sends the web page back.

Hosting

- Web hosting services work by maintaining stable and secure storage spaces.
- While web hosts provide more than just simple data storage, it's a core part of their functionality.
- Hosts store data on hardware called web servers, which allows for easy maintenance and access by online users.

The cloud

Servers accessed over the Internet that offer a range of services including:

1. Storing and accessing data and files – Uses need less storage on their computers.
2. Running cloud applications – Users can access software without needing it on their own computer.
3. Providing increased processing power – Users don’t need to have expensive hardware in their computer.

J277 - 1.3 Computer networks, connections and protocols

Web servers, clients and the cloud

The relationship between servers and clients on the Internet:

Files, software, user profiles and passwords are all stored centrally on the server.

Client sends a request to the server.

Server processes the request and responds.

Web server

In this example the mobile phone is the client. The user of the phone could be requesting a webpage via a browser, this request reaches the web server which returns the page for rendering on the mobile phone (client).

Client

File server

In this example a desktop computer is the client. The user of the computer could be requesting a file being held on a file server in a cloud storage data centre. The file server returns the file to the requesting computing (client).

The cloud

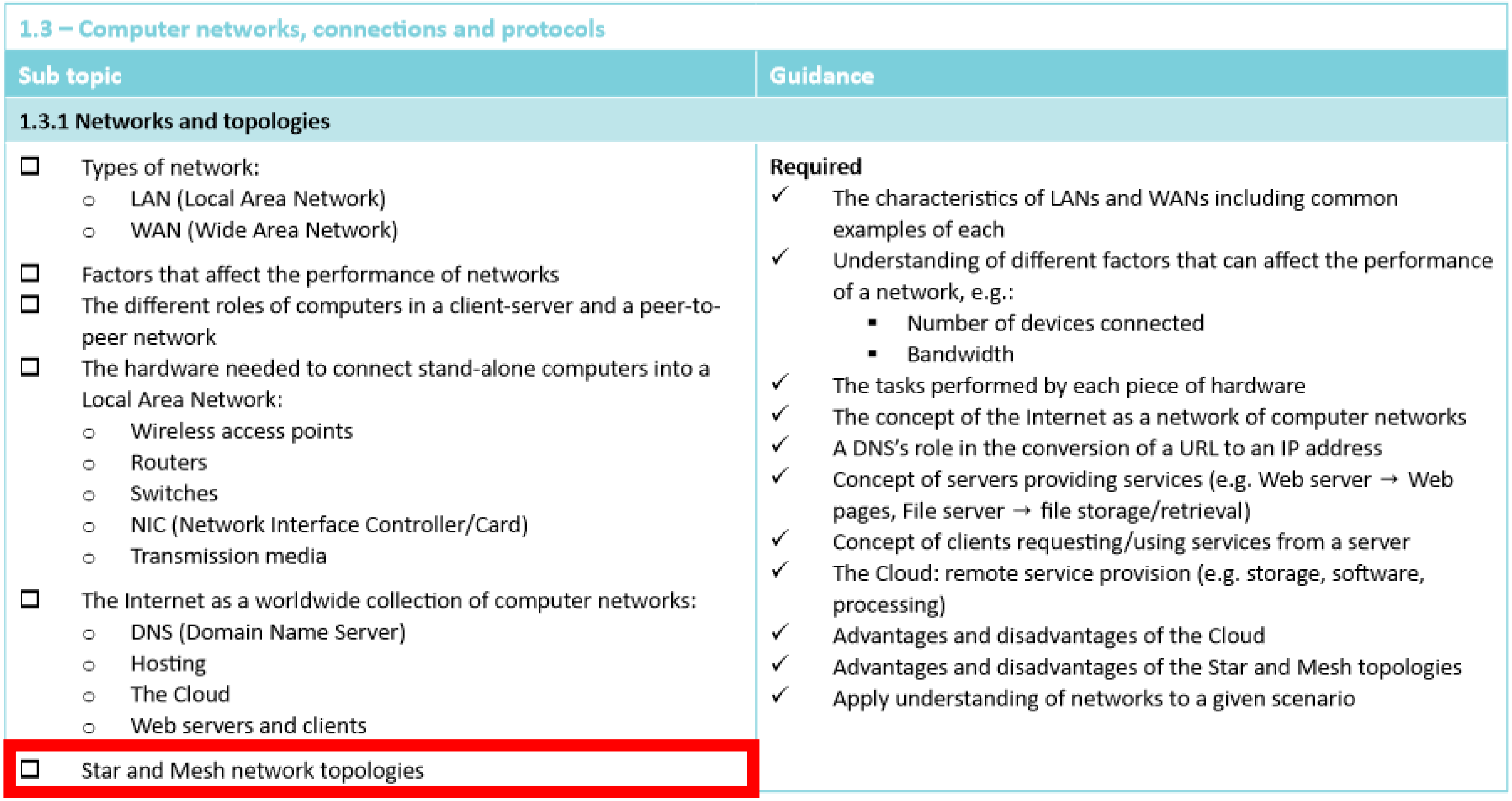
Pros:

- Access files on any device.
- No need for expensive hardware & staff as everything is hosted in the cloud.
- Excellent backup, security and recovery options.
- Easy to scale up and expand.

Cons:

- Relies on host for backups and security.
- Bandwidth issues can depend on speed of connection.
- Ongoing subscription fees for hosting data and information.
- Relies completely on a stable internet connection.

J277 - 1.3 Computer networks, connections and protocols



1.3 – Computer networks, connections and protocols

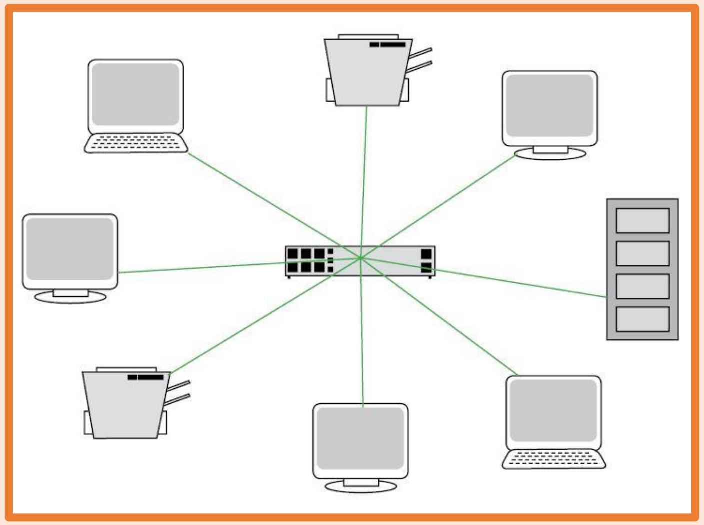
Sub topic	Guidance
1.3.1 Networks and topologies	
☐ Types of network: ○ LAN (Local Area Network) ○ WAN (Wide Area Network) ☐ Factors that affect the performance of networks ☐ The different roles of computers in a client-server and a peer-to-peer network ☐ The hardware needed to connect stand-alone computers into a Local Area Network: ○ Wireless access points ○ Routers ○ Switches ○ NIC (Network Interface Controller/Card) ○ Transmission media ☐ The Internet as a worldwide collection of computer networks: ○ DNS (Domain Name Server) ○ Hosting ○ The Cloud ○ Web servers and clients	**Required** ✓ The characteristics of LANs and WANs including common examples of each ✓ Understanding of different factors that can affect the performance of a network, e.g.: ▪ Number of devices connected ▪ Bandwidth ✓ The tasks performed by each piece of hardware ✓ The concept of the Internet as a network of computer networks ✓ A DNS's role in the conversion of a URL to an IP address ✓ Concept of servers providing services (e.g. Web server → Web pages, File server → file storage/retrieval) ✓ Concept of clients requesting/using services from a server ✓ The Cloud: remote service provision (e.g. storage, software, processing) ✓ Advantages and disadvantages of the Cloud ✓ Advantages and disadvantages of the Star and Mesh topologies ✓ Apply understanding of networks to a given scenario
☐ Star and Mesh network topologies	

J277 - 1.3 Computer networks, connections and protocols

Star and Mesh Topologies

Star Topology

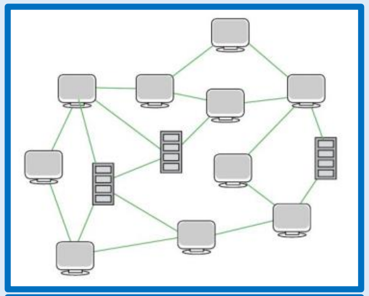
In a star topology, all of the devices connect to a central device – usually a router or switch.

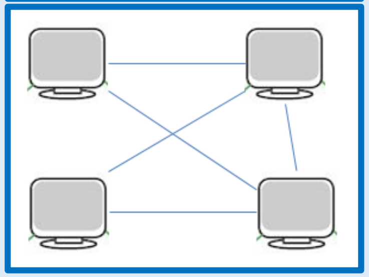


- A **star network**:
  - needs fewer cables
  - is easier to add or remove devices
  - is simple to understand and troubleshoot.
- But:
  - if the central switch fails, the whole network fails
  - causes a bottleneck in a busy network.

Mesh Topology

In a mesh topology, devices are connected to lots of other devices, with no central switch.





In a partial mesh, most devices are connected to several other devices.
In a full mesh, **every** device is connected to **all** other devices.

- A **mesh network**:
  - allows packets to be routed around bottlenecks
  - as more reliable as a single failure won't stop the rest of the network.
- But:
  - needs more cabling
  - is more complicated to add/remove devices
  - is more complicated to understand and troubleshoot.

J277 - 1.3 Computer networks, connections and protocols

1.3.2 Wired and wireless networks, protocols and layers

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

Required

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

Not required

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

Modes of connection: Wired - Ethernet

Computers can be connected to networks using many different types of wires to transmit data.

The most common type of cable in a LAN is Ethernet. This uses wires to carry electrical signals between devices and is common in most offices and classrooms.

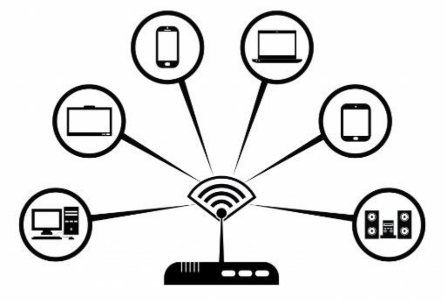
Fibre optic cables use beams of light instead of electrical signals and are used for high speed connections between switches and in many WANs.

+ Ethernet networks are more secure, as you need physical access to the cables to get into the network.
+ The connections are also more stable, faster and are less susceptible to interference.

- Ethernet networks require physical cables to be connected – this makes it much harder to change or move around.
- Cables can also be trip hazards, so should always be routed along walls, under floors or through the ceiling where possible.

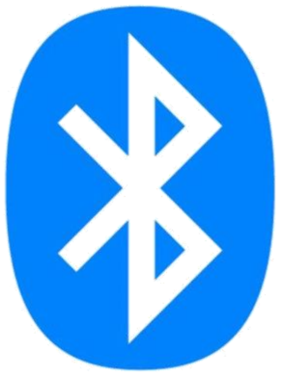
J277 - 1.3 Computer networks, connections and protocols

Modes of connection: Wireless – Wi Fi and Bluetooth



Max range: 40-100 metres.

- Wireless connections use radio waves to transmit data through the air.
- The most common type of wireless used in LANs is Wi-Fi. Devices communicate with a wireless access point (WAP) – which can be a standalone device or built into a router or switch.
- Many people have one of these as the centre of their home network and many schools and offices have Wi-Fi access as well as wired Ethernet.



Max range: 10 metres.

- Direct connection between two devices on wireless networks.
- Bluetooth is also a popular wireless connection method, though this has a much shorter range than Wi-Fi and is more typically used for a direct connection between two devices.
- Bluetooth headphones, mice and keyboards are very common.

- Wi-Fi networks are vulnerable to hacking as the radio waves can be intercepted by anyone within range, even if they are outside the building.
- Walls or obstructions will reduce the signal strength, and many electrical objects such as fridges and microwaves will cause interference.
- Transfer speeds over Wi-Fi are typically slower than Ethernet.

+ The main advantage of Wi-Fi is ease of movement.
+ You also don’t need to purchase extra cables for each device and most devices come with a Wi-Fi adapter built in.

- Bluetooth signals can also be intercepted by anyone in range.
- Bluetooth only works over a very short range and transfer speeds are very slow.

+ Bluetooth is ideal for personal devices such as headphones, keyboards and mice as the small range makes it harder to intercept the data.
+ Bluetooth uses less power to send the data than Wi-Fi and is designed to quickly make ad-hoc connections.

Practice Questions

Dave has set up a new travel agency and needs you to set up a network for the shop. There will be four workstations for staff to book new holidays and take payments from clients, and each member of staff will also be issued with a tablet so they can show customers different destinations and hotels. Staff will use their PCs to phone customers and suppliers using headsets.

Describe how different types of network connection could be set up to support then new business

- The workstations should be connected using Ethernet because they won't need to be moved, they will allow for fast transfer speeds and it is more secure than Wi-Fi for sending customer data.
- The tablets should be connected using Wi-Fi so they can be moved around without cables and the signals won't have to go through walls.
- The headsets should be connected through Bluetooth so that staff don't get tangled in wires and they can connect just to their own workstation rather than needing full access to the network.

A brand new state of the art building is being designed for a large scale corporate business. It is essential that the business has a very reliable, fast and secure data transfer on its network. State whether you would recommend a wired, wireless or hybrid solution and provide reasons to back this up.

- A state of the art building will be build with modern technology as a consideration.
- As such there will be plenty of ceiling and floor crawl spaces to hide cabling required for a wired network.
- It is also stated that a very fast and secure network is required, this is another plus for a wired solution.
- That said, it is highly likely and workers / visiting clients with expect a wireless connection in a modern office…
- …so the network could be supplemented with wireless access points (WAP) in key locations such as conference rooms, assuming the appropriate steps have been taken to secure the network.

J277 - 1.3 Computer networks, connections and protocols

1.3.2 Wired and wireless networks, protocols and layers

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

Required

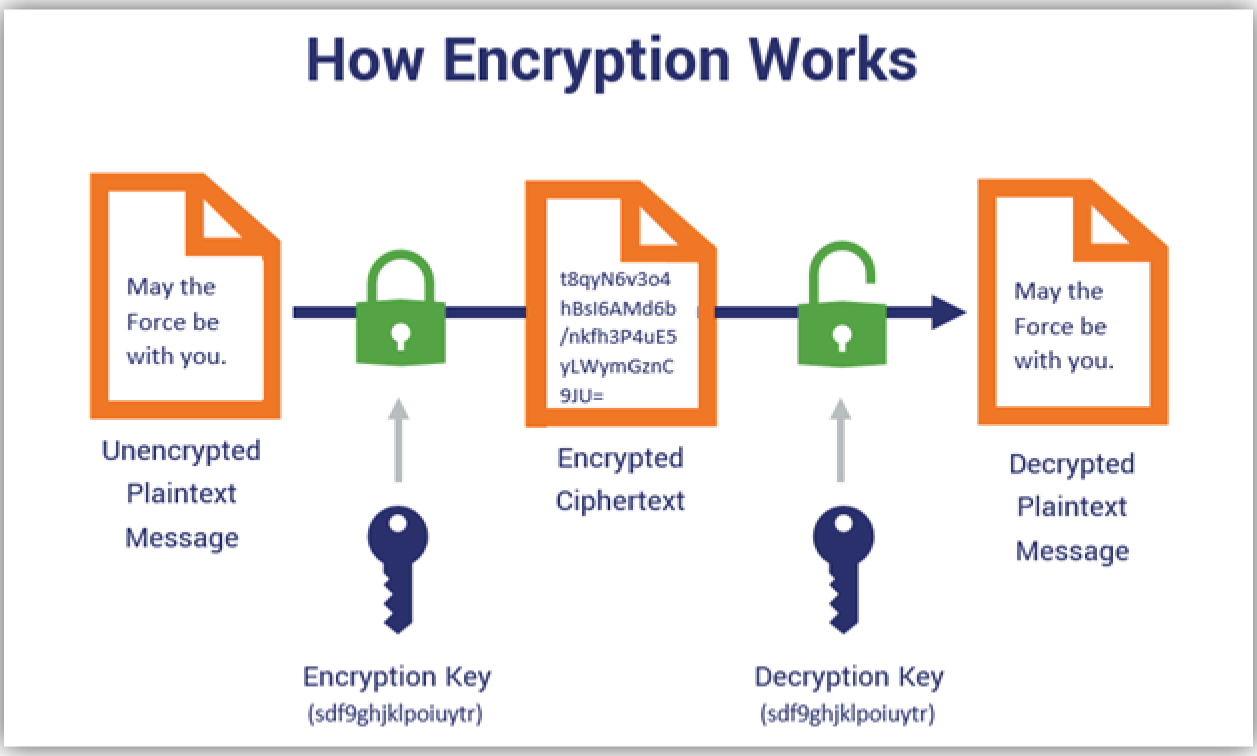
- ✓ Compare benefits and drawbacks of wired versus wireless connection
- ✓ Recommend one or more connections for a given scenario
- ✓ The principle of encryption to secure data across network connections
- ✓ IP addressing and the format of an IP address (IPv4 and IPv6)
- ✓ A MAC address is assigned to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

Not required

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

Wi-Fi encryption

- Encryption is a method of scrambling data with a key code such that it makes no sense.
- On an open or public Wi-Fi network anyone can join and sniff out packets of data from other users.
- Therefore encryption is used. If intercepted the data will have no meaning.
- In order to read the data the user is required to decrypt the data using the key.

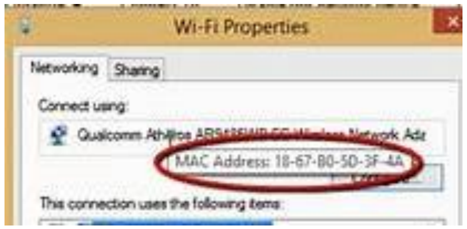

How Encryption Works
May the Force be with you.
Unencrypted Plaintext Message
t8qyN6v3o4 hBsI6AMd6b /nkfh3P4uE5 yLWymGznC 9JU=
Encrypted Ciphertext
May the Force be with you.
Decrypted Plaintext Message
Encryption Key (sdf9ghjklpoiuytr)
Decryption Key (sdf9ghjklpoiuytr)

J277 - 1.3 Computer networks, connections and protocols

The uses of IP addressing (IPv4 & IPv6) and MAC addressing

There are two types of addressing used for local and wide area networks:

MAC address	Media Access Control address	Found on network interface cards. Routes frames on a local area network between network interface cards. Every address is completely unique.
IP address	Internet Protocol address	Routes packets on a wide area network between routers.


Wi-Fi Properties

Networking | Sharing

Connect using:

Qualcomm Atheros AR9485WB-EG Wireless Network Ada

MAC Address: 18-67-B0-5D-3F-4A

This connection uses the following items:

The reason we have IPv6 is:

- Due to the shortage of IPv4 addresses, IPv6 is now used as well, though some older devices don’t support the new format.
- To solve this problem, new devices have both an IPv4 AND an IPv6 address so that they can work on any network.

IPv6 addresses are represented using 8 blocks of 4 hexadecimal digits, separated by colons
E.g. 6164:6120:6C6F:7665:6C61:6365:2043:4B4B

When displayed, an IPv6 address will often miss leading zeroes, so ***… :6365:0001:2043: …*** might appear as ***… :6365:1:2043: …***

J277 - 1.3 Computer networks, connections and protocols

1.3.2 Wired and wireless networks, protocols and layers

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

Required

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

Not required

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

J277 - 1.3 Computer networks, connections and protocols

Standards and protocols

What are standards?

An agreed set of requirements for hardware and software that allows different manufacturers to make compatible products.

What are protocols?

Rules for how devices communicate, and how data is organised and transmitted across a network.

TCP/IP	**Transmission Control Protocol / Internet Protocol**	Provides an error free transmission between two routers (TCP) and routing of packets on a wide area network (IP).
HTTP	**Hypertext Transfer Protocol**	A client-server method of requesting and delivering HTML files.
HTTPS	**Hypertext Transfer Protocol Secure**	Encryption and authentication for client-server data.
FTP	**File Transfer Protocol**	Sending files between computers. Used for uploading web pages and associated files to a web server for hosting.
POP	**Post Office Protocol**	Used by email clients to retrieve email from a mail server.
IMAP	**Internet Message Access Protocol**	Used by email clients to retrieve email from a mail server, and to manage remote mail boxes. Better than POP.
SMTP	**Simple Mail Transfer Protocol**	Sends mail to a mail server.

J277 - 1.3 Computer networks, connections and protocols

1.3.2 Wired and wireless networks, protocols and layers

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

Required

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

Not required

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

J277 - 1.3 Computer networks, connections and protocols

The concept of layers

In networking, layering means to break up the sending of messages into separate components and activities.

Each component handles a different part of the communication.

This can be referred to as the Transmission Control Protocol/Internet Protocol (TCP/IP) model.

+ Reducing the complexity of the problem.

+ Devices can be manufactured to operate at a particular layer.

+ Products from different vendors will work together.

There are four layers to be considered:

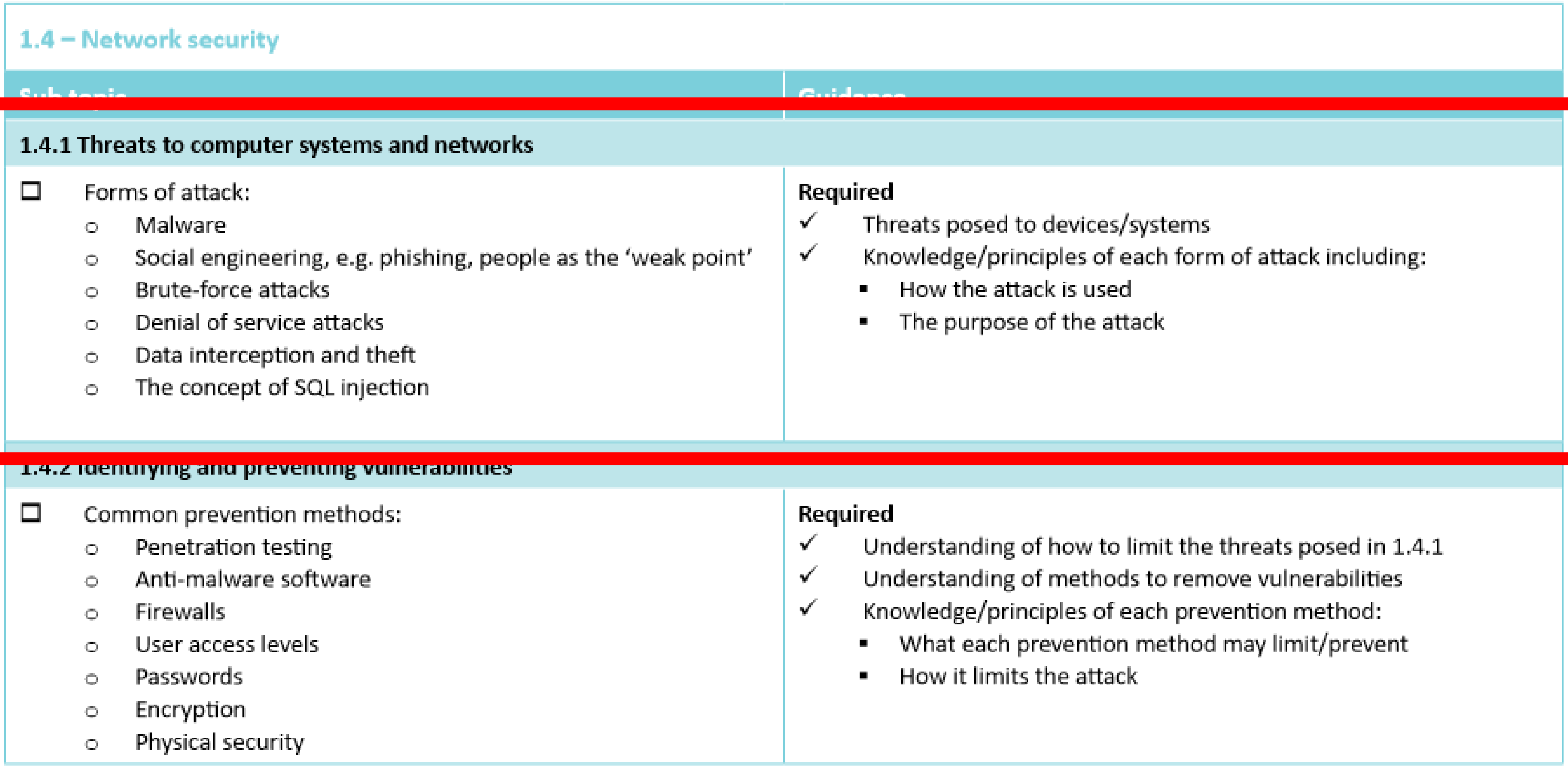
- **Application layer** - encodes/decodes the message in a form that is understood by the sender and the recipient.

- **Transport layer** - breaks down the message into small chunks (packets). Each packet is given a packet number and the total number of packets. The recipient uses this information to assemble the packets together in the correct order. It also allows the recipient to see if there are any missing packets.

- **Network layer** - adds the sender's IP address and that of the recipient. The network then knows where to send the message, and where it came from.

- **Link layer** - enables the transfer of packets between nodes on a network, and between one network and another.

J277 - 1.4 Network Security

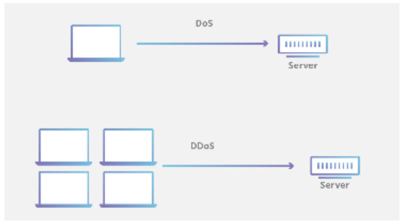
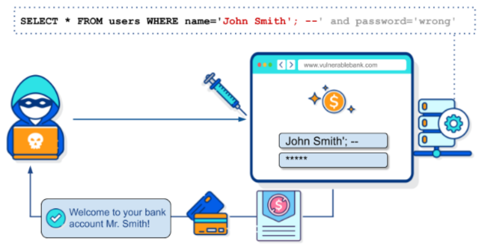


1.4 – Network security

Sub-topic	Guidance
1.4.1 Threats to computer systems and networks	
☐ Forms of attack: ○ Malware ○ Social engineering, e.g. phishing, people as the 'weak point' ○ Brute-force attacks ○ Denial of service attacks ○ Data interception and theft ○ The concept of SQL injection	**Required** ✓ Threats posed to devices/systems ✓ Knowledge/principles of each form of attack including: ▪ How the attack is used ▪ The purpose of the attack
1.4.2 Identifying and preventing vulnerabilities	
☐ Common prevention methods: ○ Penetration testing ○ Anti-malware software ○ Firewalls ○ User access levels ○ Passwords ○ Encryption ○ Physical security	**Required** ✓ Understanding of how to limit the threats posed in 1.4.1 ✓ Understanding of methods to remove vulnerabilities ✓ Knowledge/principles of each prevention method: ▪ What each prevention method may limit/prevent ▪ How it limits the attack

J277 - 1.4 Network Security

Forms of Attack

	Attack	How it works
	Data interception and theft	Data theft refers to any way sensitive information is compromised, whereas data interception is a specific type of data theft, referring to information that is captured during transmission.
	Brute- force attack	A brute force attack is a hacking method that uses trial and error to crack passwords, login credentials, and encryption keys.
	**Denial of service (DoS)**	These attacks are characterised by an explicit attempt by attackers to prevent legitimate use of a service by flooding it with useless traffic/requests.
 `SELECT * FROM users WHERE name='John Smith'; --' and password='wrong'` John Smith'; -- Welcome to your bank account Mr. Smith!	**SQL Injection**	SQL injection is the placement of malicious code in SQL statements, via web page input

J277 - 1.4 Network Security

Malware

Software designed to damage or disrupt a device or network.

Spyware – Monitors user actions and sends info to the tracker.

Scareware – Tricks users into paying to fix fake problems.

Ransomware - Encrypts files. User pays for decryption key.

Viruses – Attached to other files. Only run or replicate when the file is opened.

Worms – Similar to viruses but self-replicate so spread quickly.

Trojans – Malware disguised as legitimate software. Do not replicate themselves.

Social Engineering

To gain access to networks or sensitive information by using people as a system's weak point.

Telephone –

A person is called by someone pretending to be a friend, colleague or company and is persuaded to disclose confidential information.

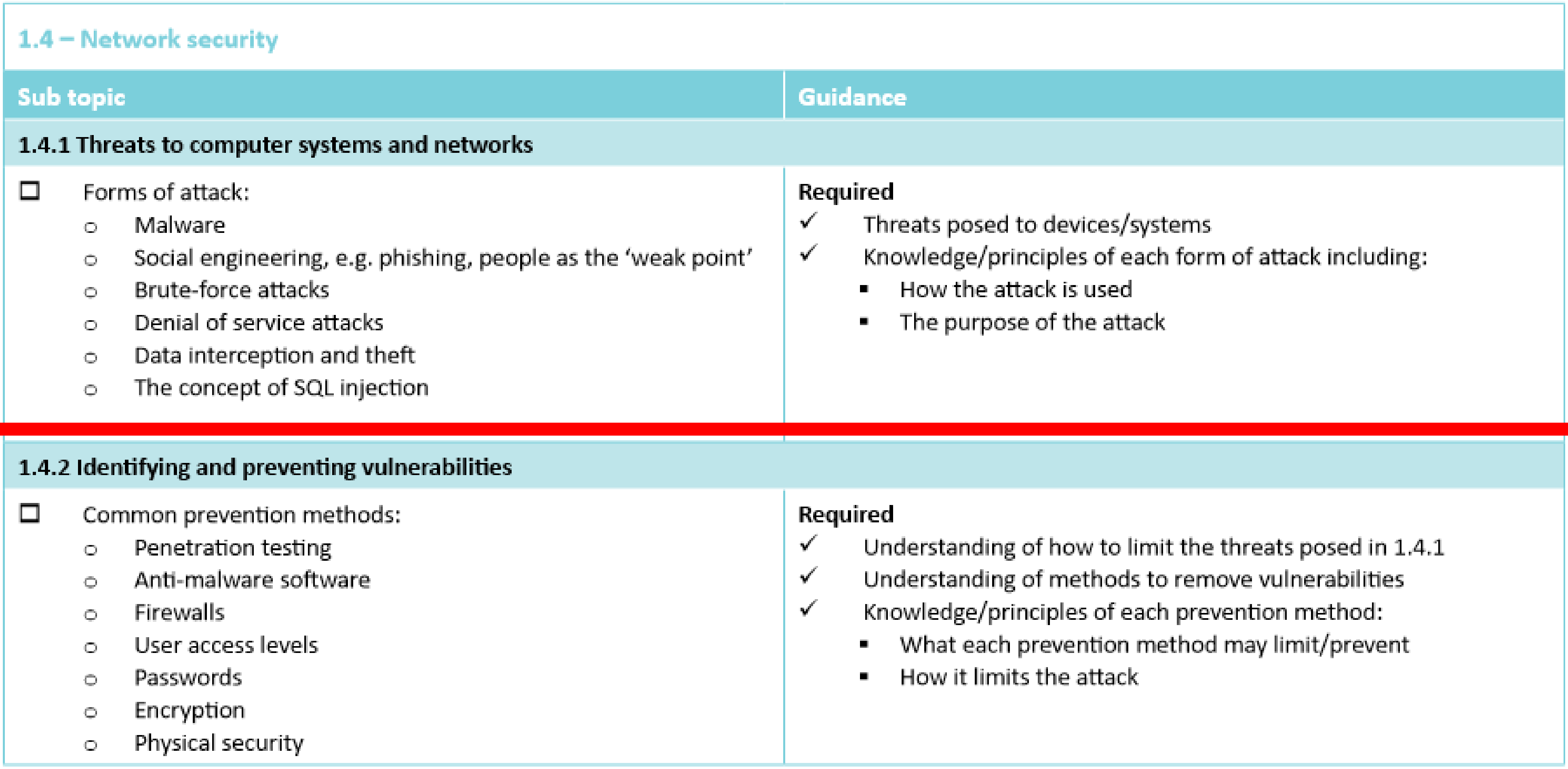
Phishing –

Criminals send emails pretending to be well-known businesses. They contain links to fake website that ask users to update their personal information, which the criminals steal.

People as the weak point –

- Not installing operating system updates.
- not keeping anti-malware up-to-date.
- Not locking doors to server/computer rooms & logging off.
- Leaving printouts with sensitive information lying around.
- Writing passwords down on sticky notes attached to computers.
- Sharing passwords.
- Using easy to guess passwords.
- Not encrypting data on portable media.
- Not having well understood, or poor network policies.
- Not training staff to protect themselves against phishing attacks.

J277 - 1.4 Network Security



1.4 – Network security

Sub topic	Guidance
1.4.1 Threats to computer systems and networks	
☐ Forms of attack: ○ Malware ○ Social engineering, e.g. phishing, people as the 'weak point' ○ Brute-force attacks ○ Denial of service attacks ○ Data interception and theft ○ The concept of SQL injection	**Required** ✓ Threats posed to devices/systems ✓ Knowledge/principles of each form of attack including: ▪ How the attack is used ▪ The purpose of the attack
1.4.2 Identifying and preventing vulnerabilities	
☐ Common prevention methods: ○ Penetration testing ○ Anti-malware software ○ Firewalls ○ User access levels ○ Passwords ○ Encryption ○ Physical security	**Required** ✓ Understanding of how to limit the threats posed in 1.4.1 ✓ Understanding of methods to remove vulnerabilities ✓ Knowledge/principles of each prevention method: ▪ What each prevention method may limit/prevent ▪ How it limits the attack

J277 - 1.4 Network Security

Network Security Measures

Penetration Testing

Tests performed under a controlled environment by a qualified person.

Checks for current vulnerabilities and explores potential ones in order to expose weaknesses in the system so they cannot be maliciously exploited.

Anti-Malware software

Software with the aim of preventing malware from entering the system.

Examples include viruses, worms and Trojan horses.

Firewalls

Software that performs a 'barrier' between a potential attacker and the computer system by examining all the data entering and leaving a network.

Identify threats using a set of security rules, blocking unauthorised access.

User Access Levels

Allows a system administrator to set up a hierarchy of users. Lower level users would have access to limited information and settings.

Higher level users can access the most sensitive data on the system.

J277 - 1.4 Network Security

Network Security Measures

Passwords

A string of characters used to gain access to a service or system, and prevent unauthorised access.

They should be strong and changed regularly to protect against brute-force attacks.

Encryption

Where data is translated into code so that only authorised users, or users with the key can
decrypt it.

Users must need the key in order to decrypt the coded file.

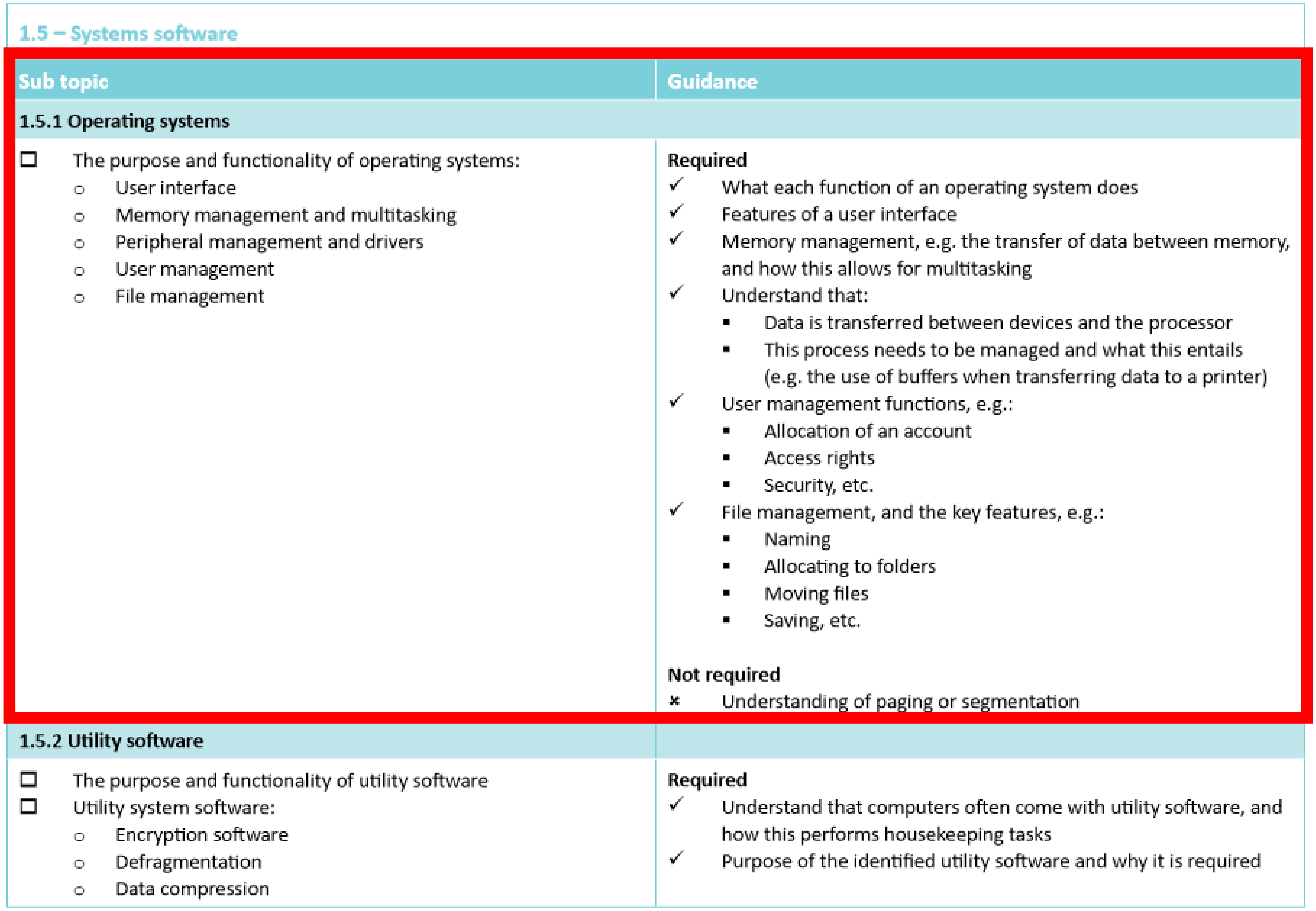
Physical Security

Physical security is used to prevent physical access to devices, and to prevent theft.

Steps may include:

door locks
window locks or bars
intruder alarm systems
CCTV systems
laptop locks security guards

J277 - 1.5 Systems software



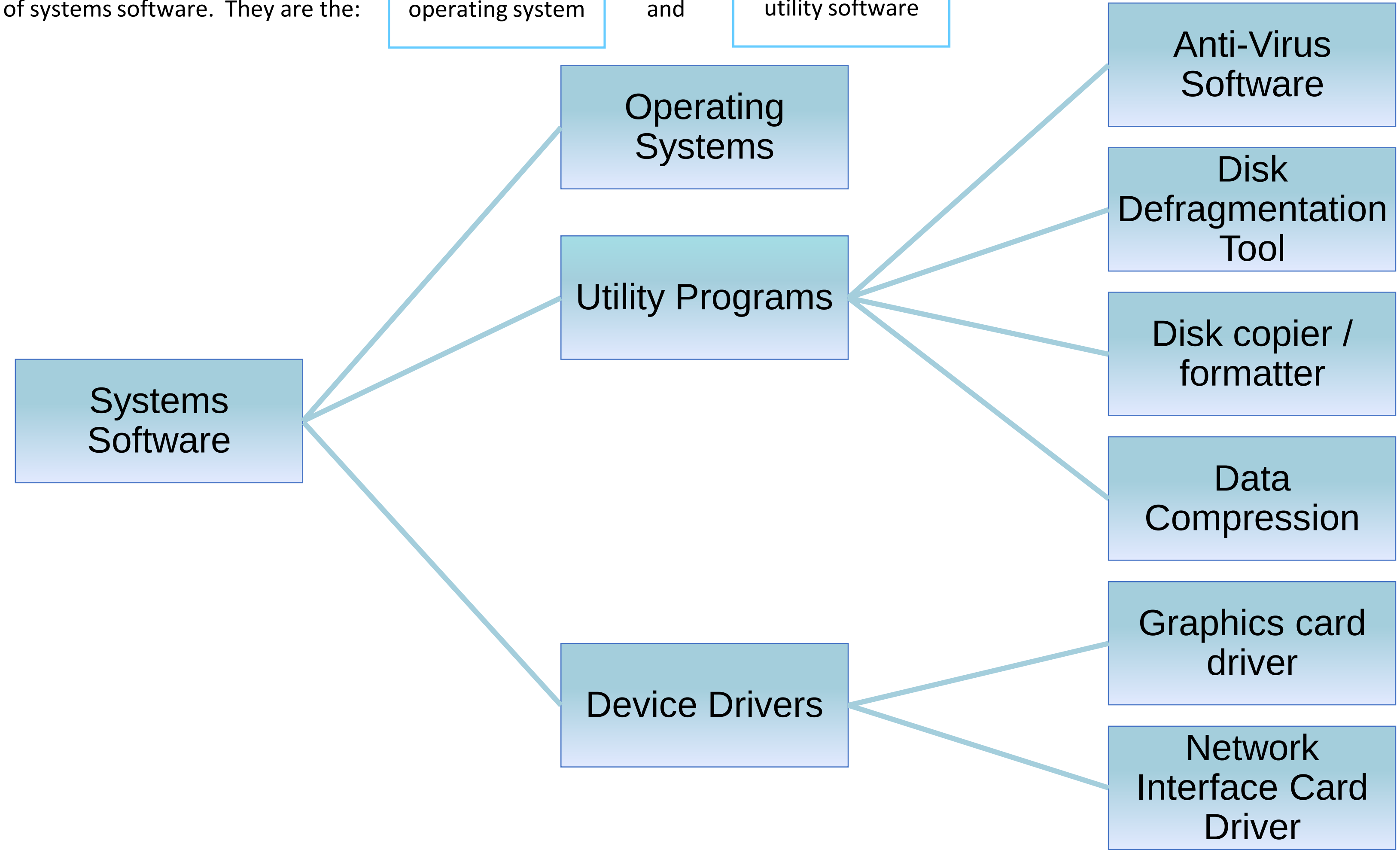
1.5 – Systems software

Sub topic	Guidance
1.5.1 Operating systems	
☐ The purpose and functionality of operating systems: ○ User interface ○ Memory management and multitasking ○ Peripheral management and drivers ○ User management ○ File management	**Required** ✓ What each function of an operating system does ✓ Features of a user interface ✓ Memory management, e.g. the transfer of data between memory, and how this allows for multitasking ✓ Understand that: ▪ Data is transferred between devices and the processor ▪ This process needs to be managed and what this entails (e.g. the use of buffers when transferring data to a printer) ✓ User management functions, e.g.: ▪ Allocation of an account ▪ Access rights ▪ Security, etc. ✓ File management, and the key features, e.g.: ▪ Naming ▪ Allocating to folders ▪ Moving files ▪ Saving, etc. **Not required** ✗ Understanding of paging or segmentation
1.5.2 Utility software	
☐ The purpose and functionality of utility software ☐ Utility system software: ○ Encryption software ○ Defragmentation ○ Data compression	**Required** ✓ Understand that computers often come with utility software, and how this performs housekeeping tasks ✓ Purpose of the identified utility software and why it is required

The purpose and functionality of systems software

The purpose of the systems software is to: provide a platform on which users can run programs to accomplish tasks, and maintain the computer system.

There are two types of systems software. They are the: operating system and utility software



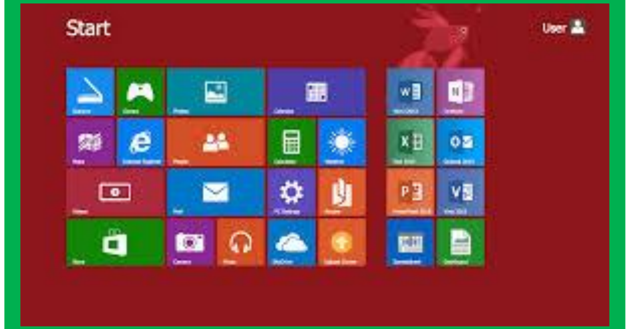
J277 - 1.5 Systems software

User Interface

The user interacts with the computer system through Command Line Interface or Graphical User Interface.

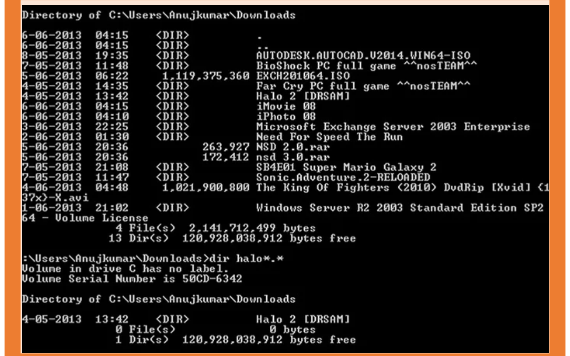
Graphical user interface (GUI):

- Windows
- Icons
- Menus
- Pointers
- Easy to use
- Visual
- Intuitive
- Optimised for mouse or touch gesture input.



Command-Line Interfaces

- Text based.
- Less resource heavy than a GUI.
- For advanced users.
- Efficient.
- More commands than a GUI.
- Automate processes using scripts.
- E.g. DOS, Raspbian (for Raspberry Pi)



```
Directory of C:\Users\Anujkumar\Downloads

6-06-2013  04:15    <DIR>          .
6-06-2013  04:15    <DIR>          ..
8-05-2013  19:35    <DIR>          AUTODESK.AUTOCAD.V2014.WIN64-ISO
7-05-2013  11:48    <DIR>          BioShock PC full game ^^nosTEAM^^
5-06-2013  06:22     1,119,375,360 EXCH201064.ISO
4-05-2013  14:35    <DIR>          Far Cry PC full game ^^nosTEAM^^
4-05-2013  13:42    <DIR>          Halo 2 [DRSAM]
6-06-2013  04:15    <DIR>          iMovie 08
6-06-2013  04:10    <DIR>          iPhoto 08
3-06-2013  22:25    <DIR>          Microsoft Exchange Server 2003 Enterprise
2-06-2013  01:30    <DIR>          Need For Speed The Run
5-06-2013  20:36           263,927 NSD 2.0.rar
5-06-2013  20:36           172,412 nsd 3.0.rar
7-05-2013  21:08    <DIR>          SB4E01 Super Mario Galaxy 2
7-05-2013  11:47    <DIR>          Sonic.Adventure.2-RELOADED
4-06-2013  04:48     1,021,900,800 The King Of Fighters (2010) DvdRip [Xvid] (1
37x)-X.avi
1-06-2013  21:02    <DIR>          Windows Server R2 2003 Standard Edition SP2
64 - Volume License
               4 File(s)  2,141,712,499 bytes
              13 Dir(s)  120,928,038,912 bytes free

:\Users\Anujkumar\Downloads>dir halo*.*
Volume in drive C has no label.
Volume Serial Number is 50CD-6342

Directory of C:\Users\Anujkumar\Downloads

4-05-2013  13:42    <DIR>          Halo 2 [DRSAM]
               0 File(s)              0 bytes
               1 Dir(s)  120,928,038,912 bytes free
```

J277 - 1.5 Systems software

Memory management: multi-tasking

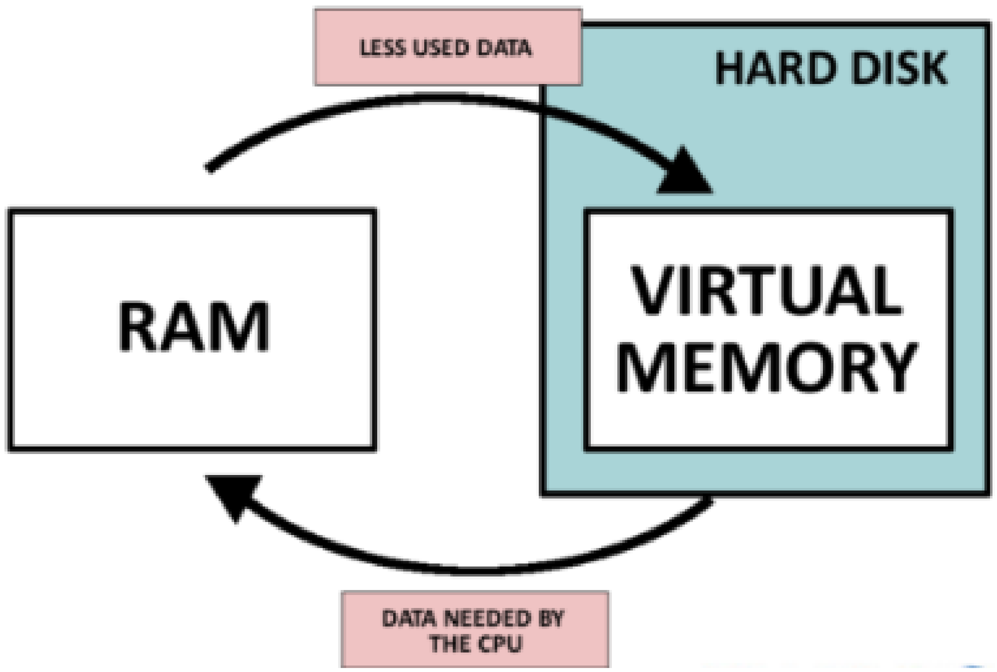
Multi-tasking is:

running multiple applications at the same time by giving each a small time-slice of processor time.

The OS allocates memory between the different programs that are open at the same time

Programmers and users do not need to know where in memory data is being held – it is the purpose of the Operating System to do this

Memory manager moves programs between virtual memory and RAM



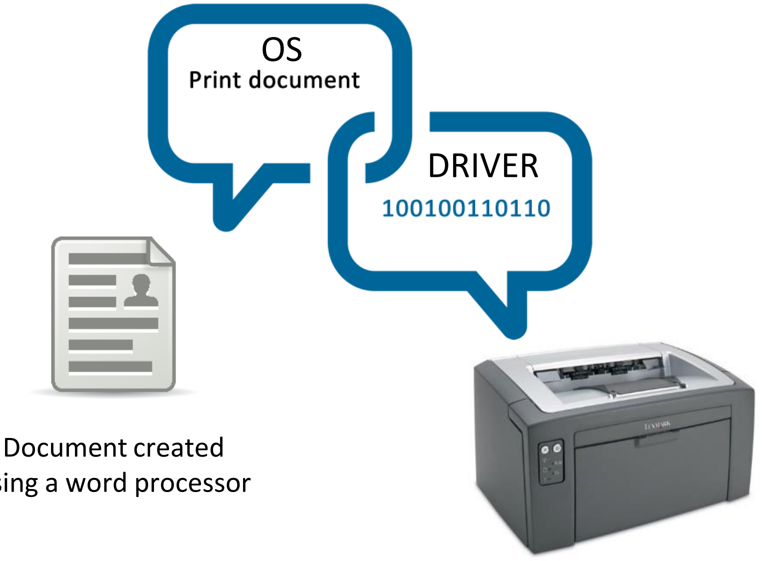
J277 - 1.5 Systems software

Peripheral management and drivers

Operating Systems manage the way in which hardware interacts with software.

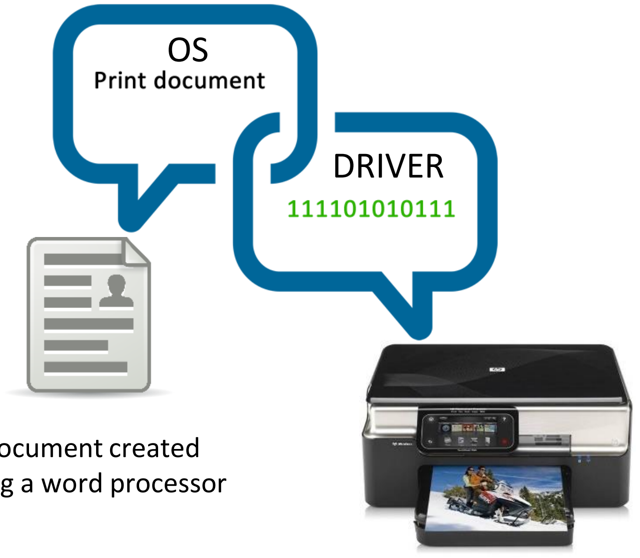
A peripheral is a piece of hardware that is not directly connected to the CPU – such as a keyboard, a mouse or even a hard disk drive.

An Operating System managed all of the peripheral devices that are connected to the computer – this allows them to be disabled, or drivers be updated.



Document created using a word processor

A laser printer attracts toner particles to magnetised areas of the paper.
Heat is used to fuse the toner to the page.



Document created using a word processor

An Inkjet printer sprays different coloured inks onto the page one row at a time.

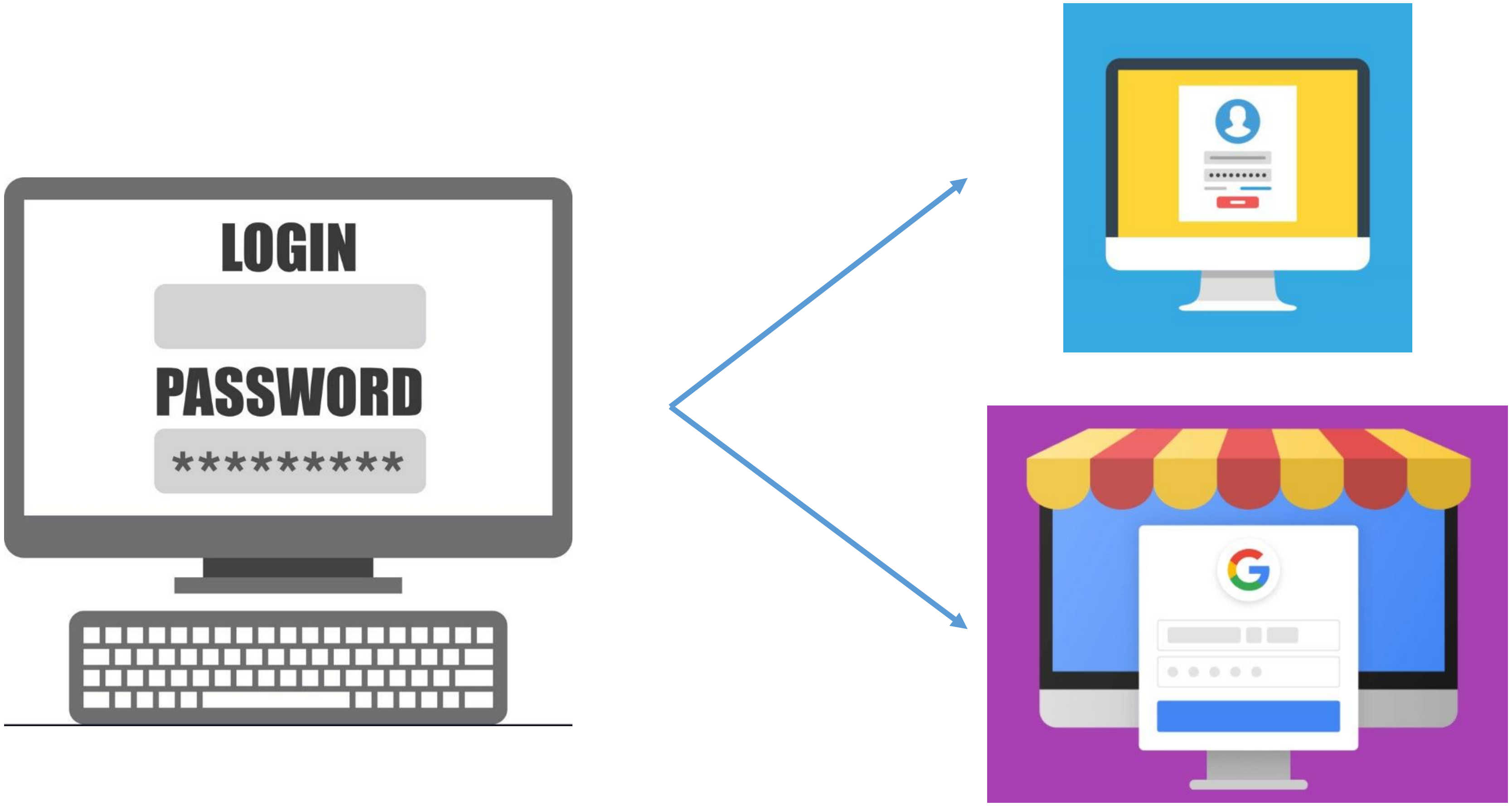
J277 - 1.5 Systems software

User management

Controls which users, and how many users, can access the computer system.

Grants users access to specific data and resource eg. Their own personal data and account, but not that of other users.

Uses anti-theft measures to prevent access for other users eg. Password or pin protection.



J277 - 1.5 Systems software

File & Disk management

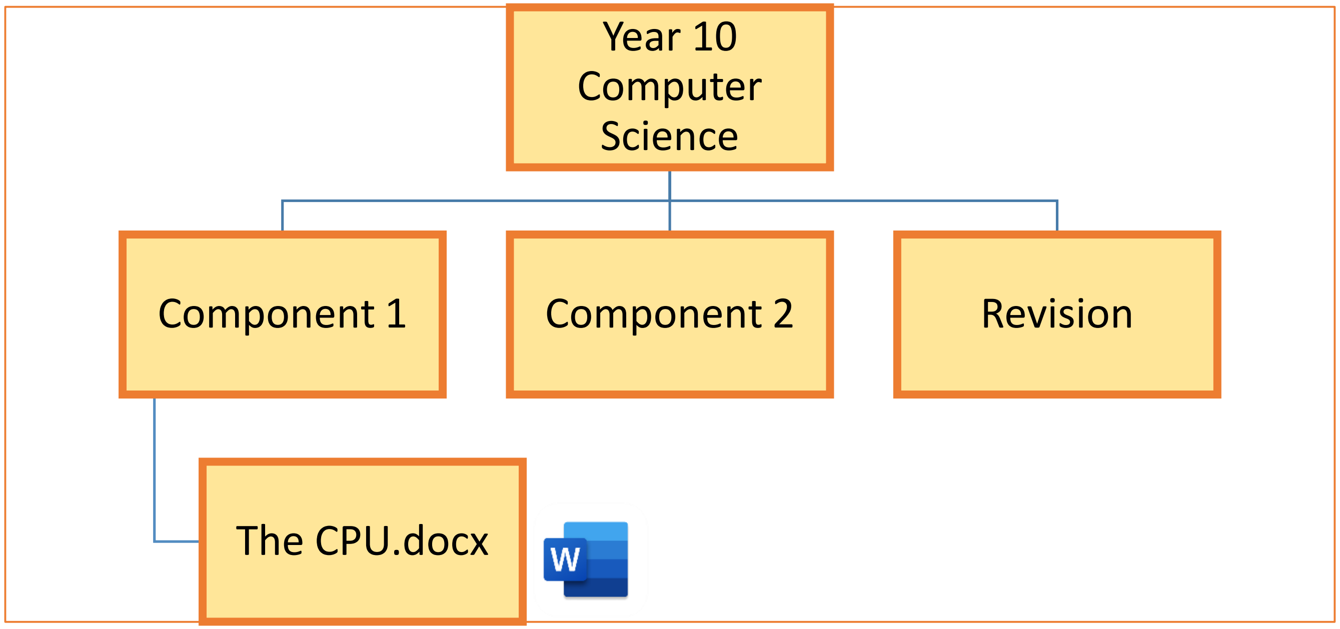
Like memory management, the Operating System must identify where files are stored for long term storage on for e.g. the hard disk drive or a solid state drive.

Organises files into hierarchical structure of folders.

Deals with naming, saving, moving, editing and deleting files and folders.

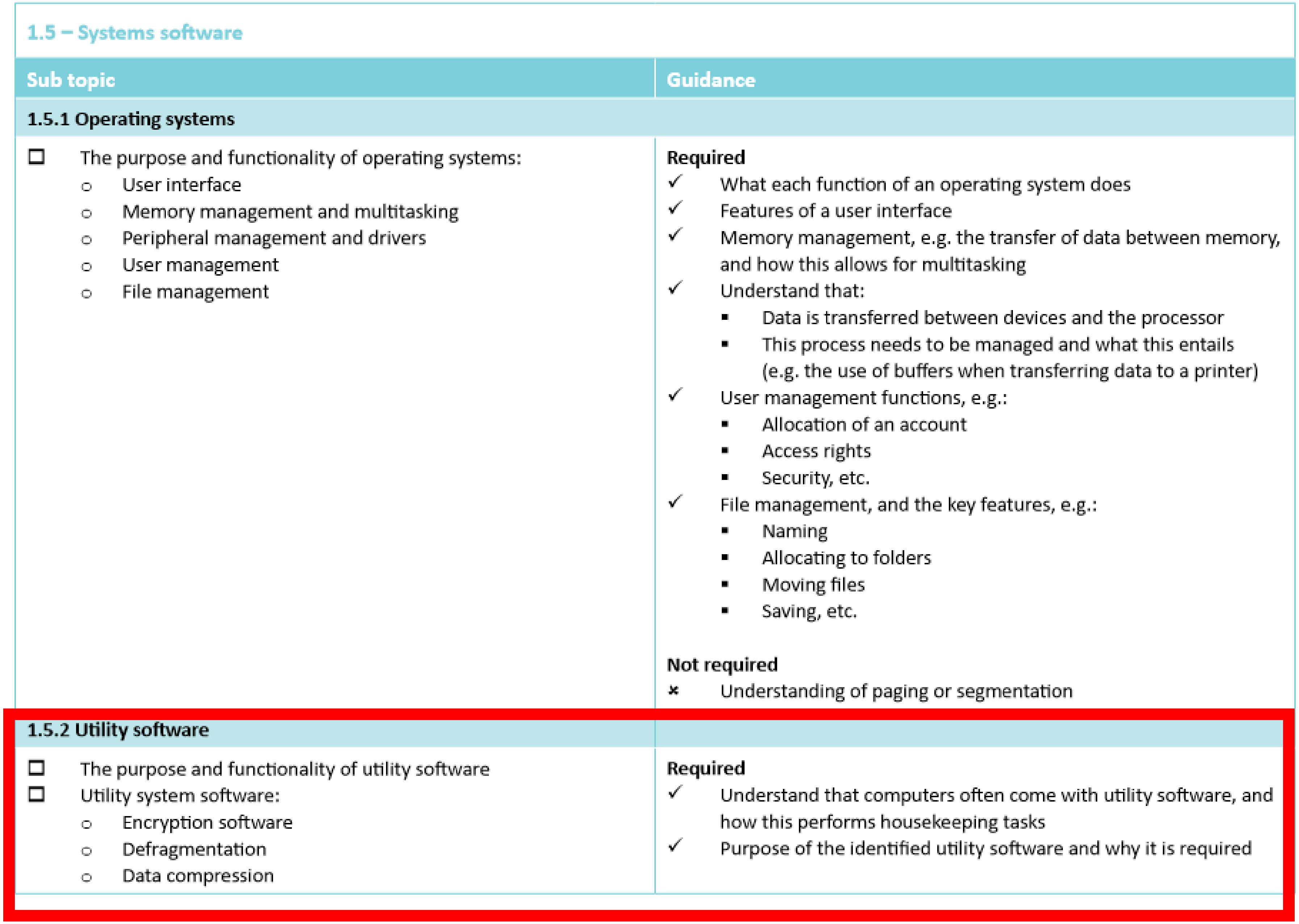
Splits the hardisk into sectors and decides where files are written to.

Maintains the hard disk with utility software.



The OS uses extension to match files with apps.

J277 - 1.5 Systems software



1.5 – Systems software

Sub topic	Guidance
1.5.1 Operating systems	
☐ The purpose and functionality of operating systems: ○ User interface ○ Memory management and multitasking ○ Peripheral management and drivers ○ User management ○ File management	**Required** ✓ What each function of an operating system does ✓ Features of a user interface ✓ Memory management, e.g. the transfer of data between memory, and how this allows for multitasking ✓ Understand that: ▪ Data is transferred between devices and the processor ▪ This process needs to be managed and what this entails (e.g. the use of buffers when transferring data to a printer) ✓ User management functions, e.g.: ▪ Allocation of an account ▪ Access rights ▪ Security, etc. ✓ File management, and the key features, e.g.: ▪ Naming ▪ Allocating to folders ▪ Moving files ▪ Saving, etc. **Not required** ✗ Understanding of paging or segmentation
1.5.2 Utility software	
☐ The purpose and functionality of utility software ☐ Utility system software: ○ Encryption software ○ Defragmentation ○ Data compression	**Required** ✓ Understand that computers often come with utility software, and how this performs housekeeping tasks ✓ Purpose of the identified utility software and why it is required

J277 - 1.5 Systems software

The purpose and functionality of Utility system software

Utility:

Purpose

Encryption

Encryption means to scramble data in a way that it is unreadable to anybody who doesn't have a key to be able to unscramble this.

Data is encrypted using a key and then decrypted using a key

Data is encrypted so that it is unreadable if anyone is to intercept it

Defragmentation

When a hard disk drive is new – files get added onto the disk in order.

As files are deleted – this leaves gaps.

When new files are saved – the files fill the gaps and become fragmented.

Defragmentation software groups fragmented files back together.

Data compression

Reducing the size of the file by performing an algorithm on the original data.

Compressed files take up less disk space and ae quicker to download.

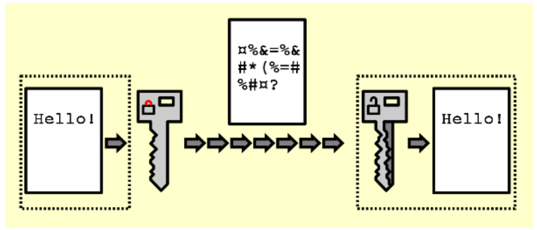
Compressed files need to be extracted before they can be used.

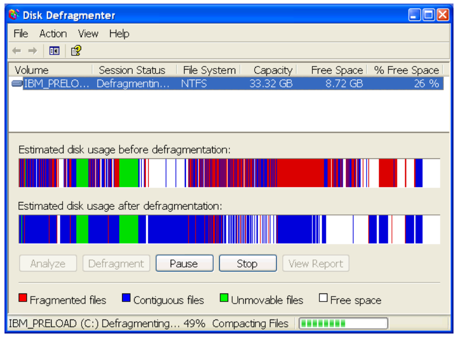
Backup

Backup software takes a copy of files to protect against data loss and malware attacks.

Backups can either be full backups where every file is copied, or incremental where only the files that have changed since the last backup are copied.

Example:





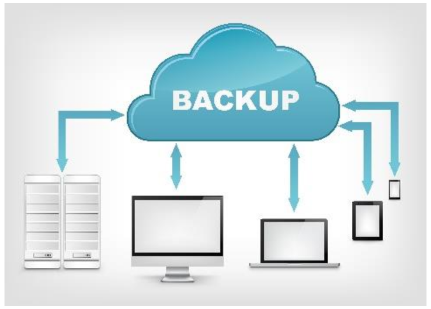
Volume	Session Status	File System	Capacity	Free Space	% Free Space
IBM_PRELO...	Defragmentin...	NTFS	33.32 GB	8.72 GB	26 %

Estimated disk usage before defragmentation:

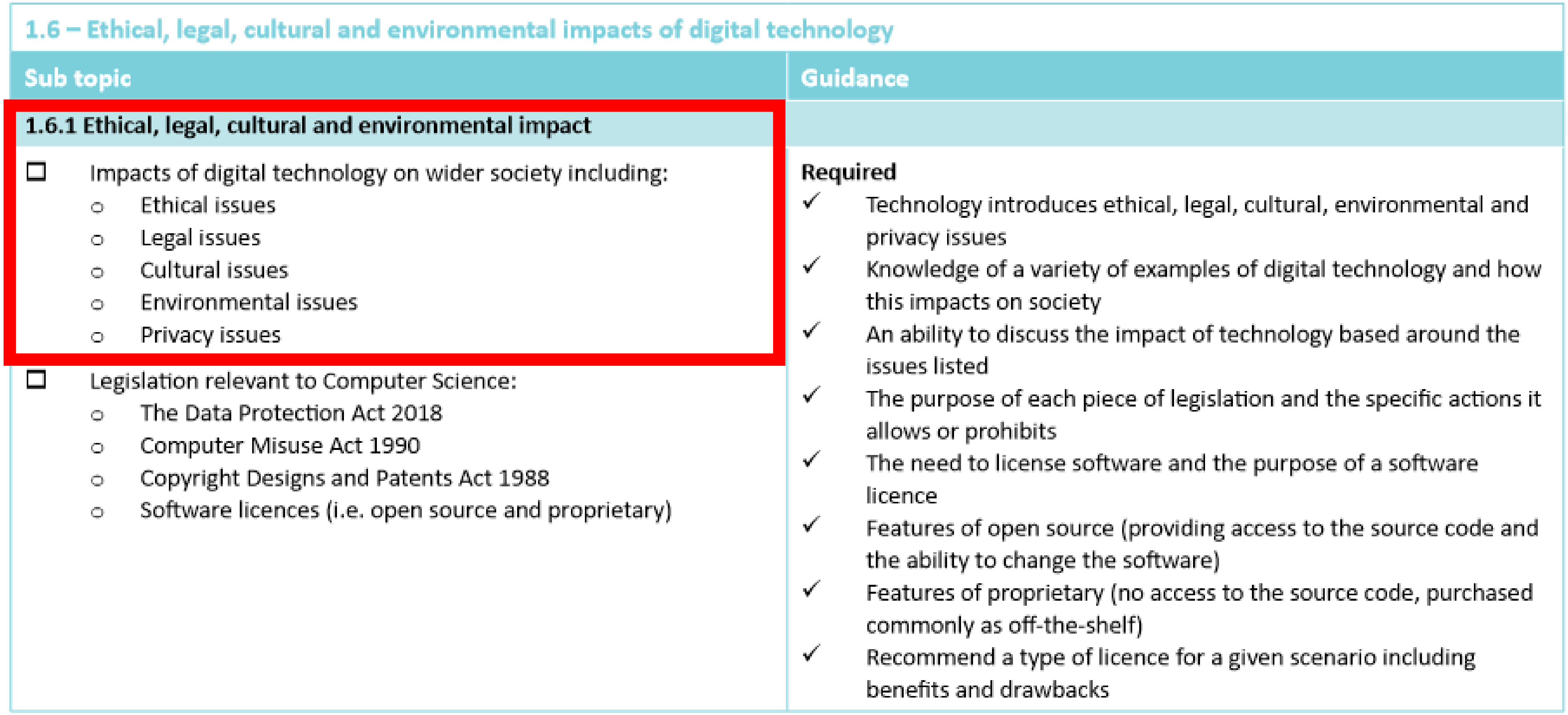
Estimated disk usage after defragmentation:

🟥 Fragmented files 🟦 Contiguous files 🟩 Unmovable files ☐ Free space

IBM_PRELOAD (C:) Defragmenting... 49% Compacting Files



J277 - 1.6 Ethical, legal, cultural and environmental impact



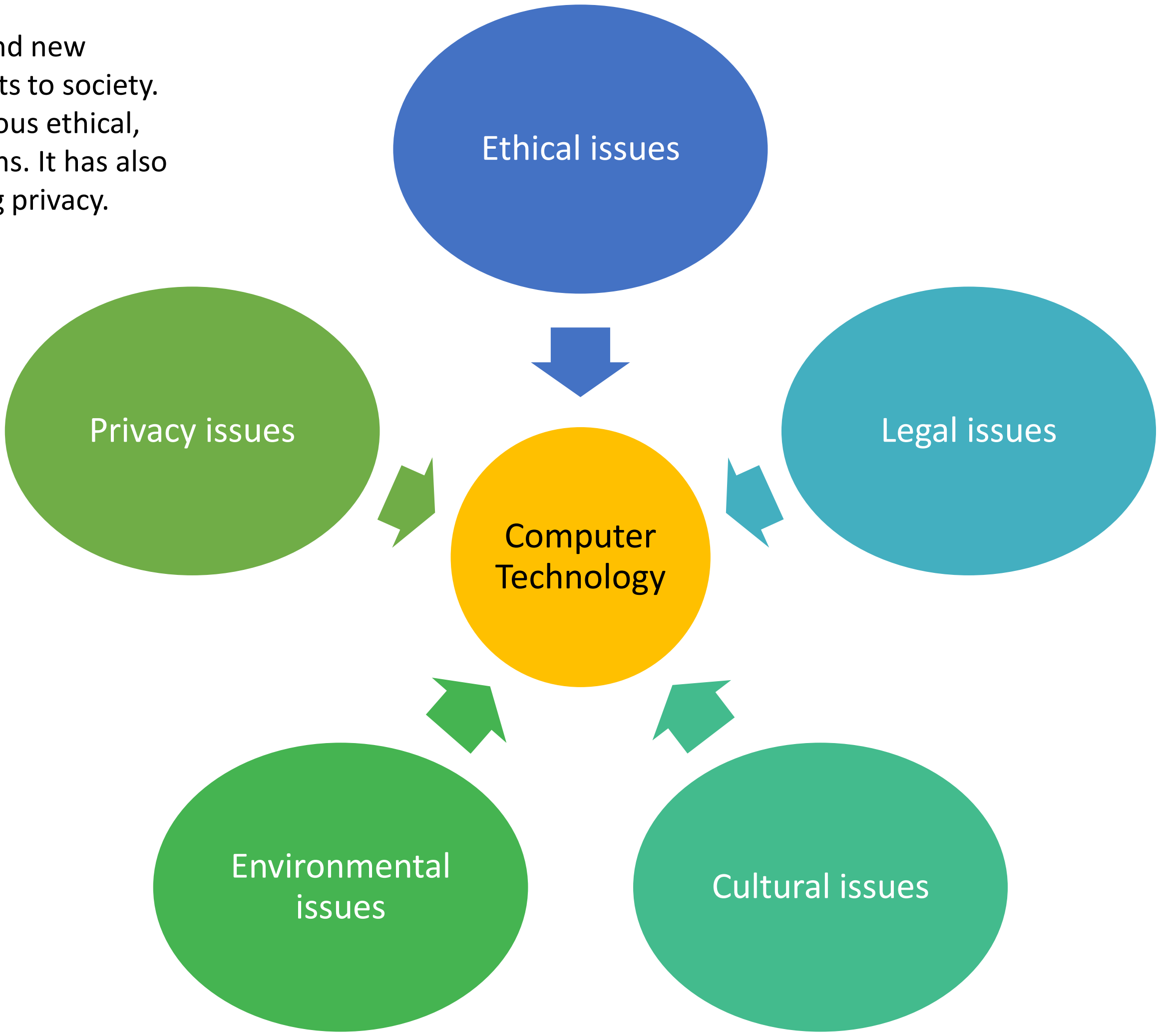
1.6 – Ethical, legal, cultural and environmental impacts of digital technology

Sub topic	Guidance
1.6.1 Ethical, legal, cultural and environmental impact	
❑ Impacts of digital technology on wider society including: ○ Ethical issues ○ Legal issues ○ Cultural issues ○ Environmental issues ○ Privacy issues ❑ Legislation relevant to Computer Science: ○ The Data Protection Act 2018 ○ Computer Misuse Act 1990 ○ Copyright Designs and Patents Act 1988 ○ Software licences (i.e. open source and proprietary)	**Required** ✓ Technology introduces ethical, legal, cultural, environmental and privacy issues ✓ Knowledge of a variety of examples of digital technology and how this impacts on society ✓ An ability to discuss the impact of technology based around the issues listed ✓ The purpose of each piece of legislation and the specific actions it allows or prohibits ✓ The need to license software and the purpose of a software licence ✓ Features of open source (providing access to the source code and the ability to change the software) ✓ Features of proprietary (no access to the source code, purchased commonly as off-the-shelf) ✓ Recommend a type of licence for a given scenario including benefits and drawbacks

J277 - 1.6 Ethical, legal, cultural and environmental impact

Impacts of Computer technologies

Computer use in the UK is widespread and new technologies have provided many benefits to society. However, this technology has raised various ethical, legal, cultural and environmental concerns. It has also highlighted important issues surrounding privacy.

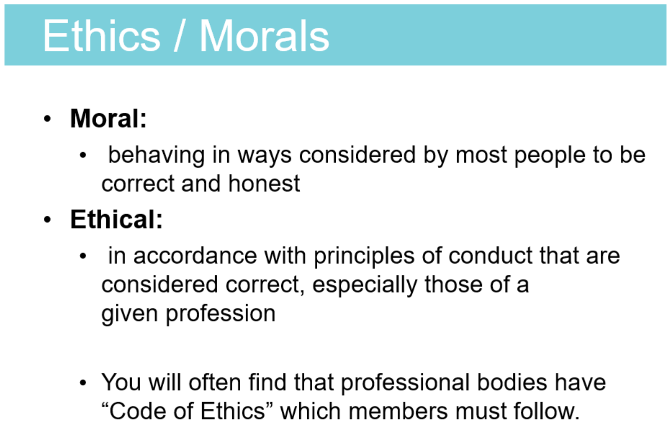


J277 - 1.6 Ethical, legal, cultural and environmental impact

Ethical issues

- **Ethics** are moral principles, or rules, which govern a person's attitudes and behaviour.

- Ethics apply to the use of computers as much as they do to other things in life. Ethical issues in computing include:
    - Ensuring public safety - the introduction of new technologies brings safety concerns. For example, driverless cars may soon be on the roads in the UK. The designers of driverless cars have not only had to ensure the safety of passengers, but also of other drivers and pedestrians.
    - Security of data - there are people that attempt to hack systems in order to gain access to other people's data. Social media accounts, phone mailboxes and networks that computers connect to are all prone to hacking.



Ethics / Morals

- **Moral:**
    - behaving in ways considered by most people to be correct and honest
- **Ethical:**
    - in accordance with principles of conduct that are considered correct, especially those of a given profession

    - You will often find that professional bodies have "Code of Ethics" which members must follow.

J277 - 1.6 Ethical, legal, cultural and environmental impact

Privacy issues

Personal information can be hard to keep private on the internet.

- Websites may ask for a name and date of birth to set up an account.
- Social media encourages users to share photos, job details etc.
- Cloud servers store personal files on their servers.

Privacy agreements say what a company can do with your information. You have to accept before using their service.

Privacy settings can sometimes be changed to make data more private. They're often fairly relaxed by default.

J277 - 1.6 Ethical, legal, cultural and environmental impact

Cultural issues

The introduction of computers has changed society, sometimes for the better, sometimes for the worse. 'Cultural issues' is the term used for computer matters that have an effect on the nature and culture of society. Some of these issues include:

The digital divide

The inequality caused by unequal access to technology.

- Devices and an internet connection can be too expensive.
- Urban areas often have great network coverage than rural areas.
- People my have difficulty adopting new technology – usually due to not being taught how to use it properly.

The changing nature of employment

Many companies now allow employees to work from home and communicate with them via technology. Cloud-based document storage enables workers to share documents with their employer, and communication can be via email or by video.

Many technology-based jobs have been moved abroad, where costs are cheaper. Many organisations have call centres/support centres in other countries. The cost of communicating with people in those countries is outweighed by the savings made.

The use of technology within the workplace has changed the nature of employment. Automation of processes using technology has led to a fall in manual, low-skilled work, such as warehouse packing. On the other hand, more high-skilled work is now available, which includes the maintenance of automated systems.

Wellbeing and Health

Eyestrain from looking at screen for too long.

Repetitive strain injury.

Back pain from poor posture or bad seating.

J277 - 1.6 Ethical, legal, cultural and environmental impact

Environmental issues

Environmental issues are those where the manufacturing and use of computers has had a negative impact on the environment.

Resources are needed to in order for computers to be produced, distributed and used. Metals and plastics are used to manufacture components, while energy is expended in distributing equipment and in using it.

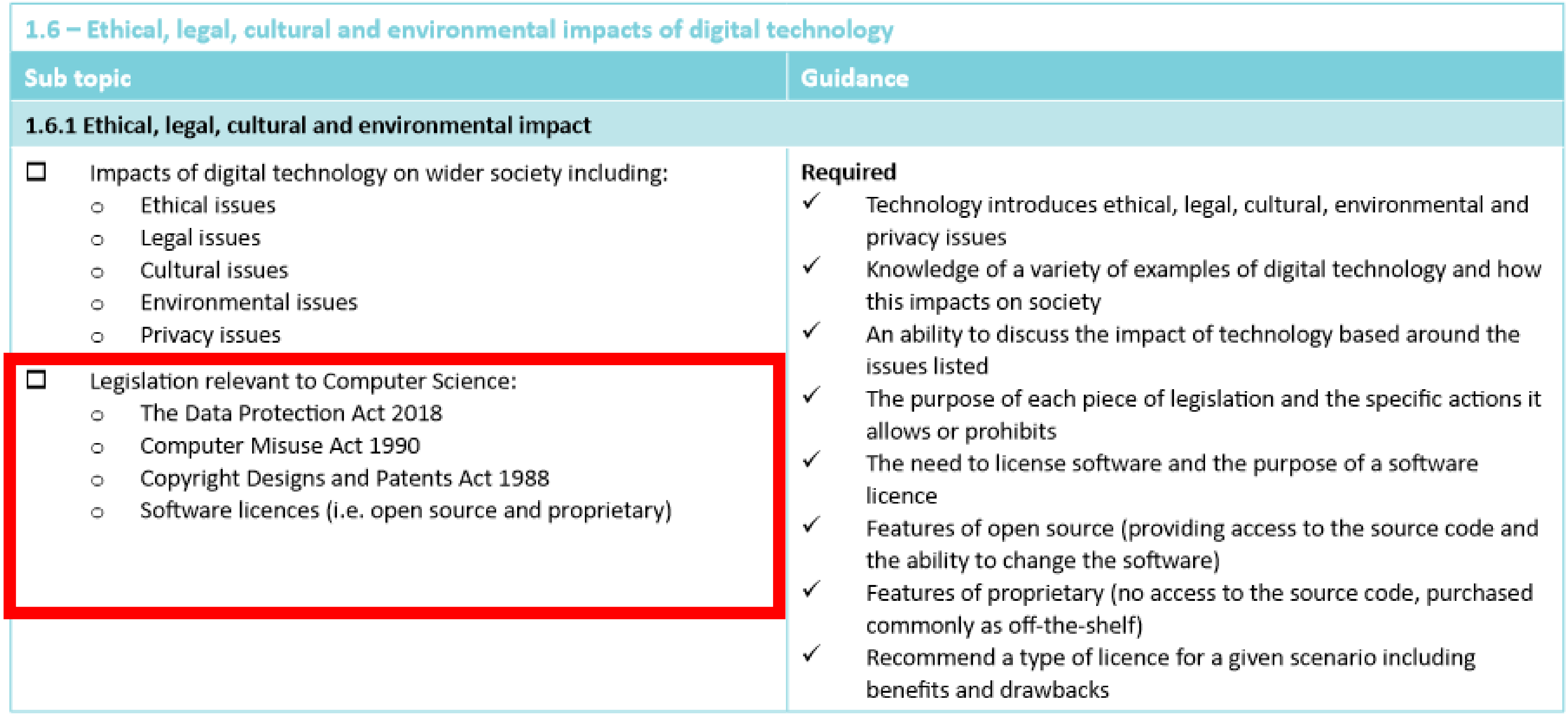
Many computers, such as **web servers**, **domain name servers** and **data centres**, need to be left running continuously. This requires lots of energy to maintain. Additionally, businesses, organisations, schools and homes all now have greater access to technology.

Many computer components are either hard to recycle or contain toxic materials, such as lead. Also, users discard ICT equipment quite quickly:

- People have new smartphones every couple of years.
- Many organisations replace computers after three or four years.
- Many people replace older technology before it fails simply because they perceive it to be old-fashioned or out of date.

All of this means that computers have a heavy impact on the environment, which is unlikely to decrease in the near future. However, many devices are now more power efficient than their predecessors and some companies have come up with innovative ways to save power.

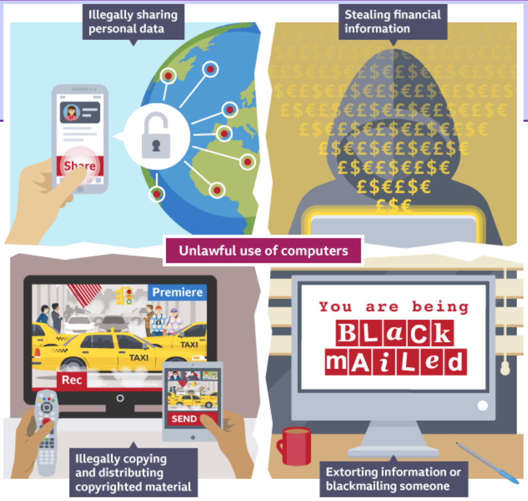
J277 - 1.6 Ethical, legal, cultural and environmental impact



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

Sub topic	Guidance
1.6.1 Ethical, legal, cultural and environmental impact	
☐ Impacts of digital technology on wider society including: ○ Ethical issues ○ Legal issues ○ Cultural issues ○ Environmental issues ○ Privacy issues ☐ Legislation relevant to Computer Science: ○ The Data Protection Act 2018 ○ Computer Misuse Act 1990 ○ Copyright Designs and Patents Act 1988 ○ Software licences (i.e. open source and proprietary)	**Required** ✓ Technology introduces ethical, legal, cultural, environmental and privacy issues ✓ Knowledge of a variety of examples of digital technology and how this impacts on society ✓ An ability to discuss the impact of technology based around the issues listed ✓ The purpose of each piece of legislation and the specific actions it allows or prohibits ✓ The need to license software and the purpose of a software licence ✓ Features of open source (providing access to the source code and the ability to change the software) ✓ Features of proprietary (no access to the source code, purchased commonly as off-the-shelf) ✓ Recommend a type of licence for a given scenario including benefits and drawbacks

J277 - 1.6 Ethical, legal, cultural and environmental impact


Illegally sharing personal data
Share
Stealing financial information
Unlawful use of computers
Premiere
TAXI
Rec
SEND
Illegally copying and distributing copyrighted material
You are being BLaCk mAiLed
Extorting information or blackmailing someone

Legal issues

Six principles of the Data Protection Act 2018:

1. Personal data must be fairly and lawfully processed.
2. Personal data must be obtained for specified, explicit and legitimate purposes.
3. Personal data must be adequate, relevant and not excessive.
4. Personal data must be accurate and up to date.
5. Personal data must not be kept for longer than is necessary.
6. Personal data must be handled in a way that ensures security.

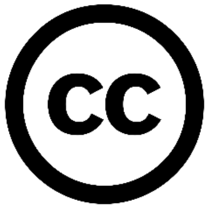
Data Protection Act
2018

Copyright Designs and Parents Act 1988:

It is illegal to copy, modify or distribute software, music, video or other intellectual property without permission of the author.

The Creative Commons Licensing:

This is a standardised way to grant copyright permissions to creative work. It allows an author to retain copyright while allowing others to copy, distribute, and make some uses of their work.



Computer Misuse Act 1990:

It is illegal to:

1. Make any unauthorised access to data,
2. with the intent to commit further offences;
3. with the intent to modify data, e.g. viruses.

Freedom of Information Act 2000:

To provide the public with access to information held by public authorities.

1. Public authorities are obliged to publish certain information about their activities.
2. Members of the public are entitled to request information from public authorities.

Open source vs proprietary software

Software may be developed to be freely distributed, or protected by copyright. This is known as open-source and closed-source (proprietary).

Open source

Open source software can be free of copyright and is usually available to anyone.

ADVANTAGES

- It costs nothing.
- **Source code** so that anyone can modify it.
- It can have many authors. So programmers can all contribute to the development of a program over time, refining and improving it and adding extra features.

DISADVANTAGES

- There is no guarantee that it works properly as there is no requirement for anyone to ensure it is **bug** free.
- Support might not be readily available, especially if the program is not in widespread use.

Proprietary software

Proprietary software is copyrighted and only available under licence.

ADVANTAGES

- The product should be free of bugs.
- Help can be sought from the organisation who supplied the software if problems occur.
- Software updates are often available, although usually at a cost.

DISADVANTAGES

- There is an initial or ongoing (subscription) cost.
- Software cannot be adapted to meet the needs of the user.
- It can be limited to a single **network**, so unless the licence allows it, a user may not redistribute the software.

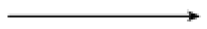
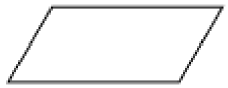
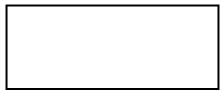
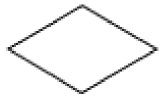
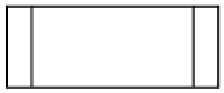
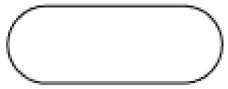
J277 - 2.1 Algorithms

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

2.1 – Algorithms

Sub topic	Guidance
2.1.1 Computational thinking	
☐ Principles of computational thinking: ○ Abstraction ○ Decomposition ○ Algorithmic thinking	**Required** ✓ Understanding of these principles and how they are used to define and refine problems
2.1.2 Designing, creating and refining algorithms	
☐ Identify the inputs, processes, and outputs for a problem ☐ Structure diagrams ☐ Create, interpret, correct, complete, and refine algorithms using: ○ Pseudocode ○ Flowcharts ○ Reference language/high-level programming language ☐ Identify common errors ☐ Trace tables	**Required** ✓ Produce simple diagrams to show: ▪ The structure of a problem ▪ Subsections and their links to other subsections ✓ Complete, write or refine an algorithm using the techniques listed ✓ Identify syntax/logic errors in code and suggest fixes ✓ Create and use trace tables to follow an algorithm

Flowchart symbols

	**Line**		**Input/ Output**
	**Process**		**Decision**
	**Sub program**		**Terminal**

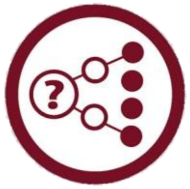
J277 - 2.1 Algorithms

Key techniques for computational thinking

Abstraction

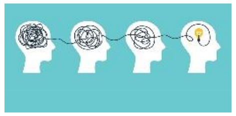
Representing 'real world' problems in a computer using variables and symbols and removing unnecessary elements from the problem.

Decomposition



Breaking down a large problem into smaller sub-problems.

Algorithmic thinking



Identifying the steps involved in solving a problem.

Example: Find the quickest route by car between two places.

Details to ignore	Details to focus on
Distance crow flies	Shortest route along the roads
Road names	Traffic information

What is the length of each route?
What are the speed limits on each route?

1. List all potential routes.
2. Find lengths of each route.
3. Calculate time for each route.
4. Find route with shortest time.

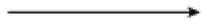
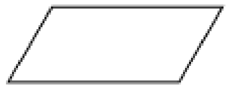
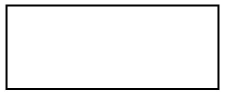
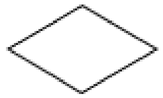
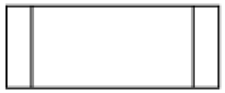
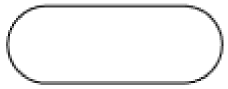
J277 - 2.1 Algorithms

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

2.1 – Algorithms

Sub topic	Guidance
2.1.1 Computational thinking	
☐ Principles of computational thinking: ○ Abstraction ○ Decomposition ○ Algorithmic thinking	**Required** ✓ Understanding of these principles and how they are used to define and refine problems
2.1.2 Designing, creating and refining algorithms	
☐ Identify the inputs, processes, and outputs for a problem ☐ Structure diagrams ☐ Create, interpret, correct, complete, and refine algorithms using: ○ Pseudocode ○ Flowcharts ○ Reference language/high-level programming language ☐ Identify common errors ☐ Trace tables	**Required** ✓ Produce simple diagrams to show: ▪ The structure of a problem ▪ Subsections and their links to other subsections ✓ Complete, write or refine an algorithm using the techniques listed ✓ Identify syntax/logic errors in code and suggest fixes ✓ Create and use trace tables to follow an algorithm

Flowchart symbols

	**Line**		**Input/ Output**
	**Process**		**Decision**
	**Sub program**		**Terminal**

J277 - 2.1 Algorithms

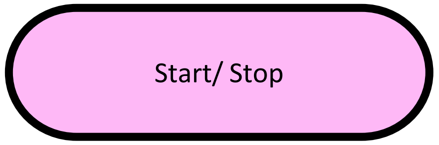
Identify the input, processes and outputs for a problem

An input is:	Any information or data which goes into a system.
A process is:	Anything which happens to information or data during a programs execution e.g. performing calculations or conversions.
An output is:	Any information of data which leaves a system.

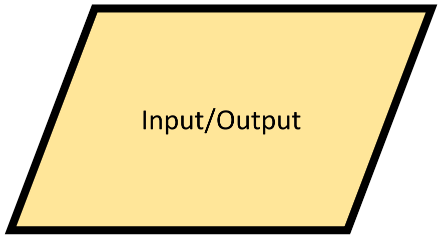
Title of program	**What does it do?**	**Inputs**	**Processes**	**Outputs**
Temperature Converter	Converts the temperature from Celsius to Fahrenheit	Temperature in Celsius (e.g., 25 degrees)	Convert the Celsius temperature to Fahrenheit	Temperature in Fahrenheit (e.g., 77 degrees)
Addition Calculator	Add 2 numbers together	Two numbers (e.g., 5 and 3)	Add the two numbers together	The sum of the two numbers (e.g., 8)
BMI Calculator	Works out a person's BMI	Person's weight (in kg) and height (in meters)	Calculate the Body Mass Index (BMI) using the weight and height	The calculated BMI value (e.g., 23.4)
File Sorter	Sorts files into alphabetical order	List of unsorted filenames (e.g., ["file3.txt", "file1.txt", "file2.txt"])	Sort the filenames alphabetically	Sorted list of filenames (e.g., ["file1.txt", "file2.txt", "file3.txt"])

J277 - 2.1 Algorithms

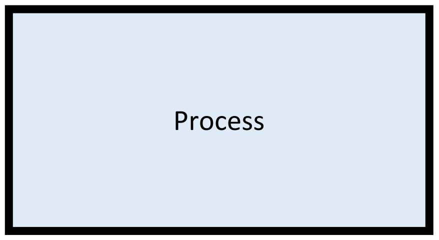
Flow diagram symbols



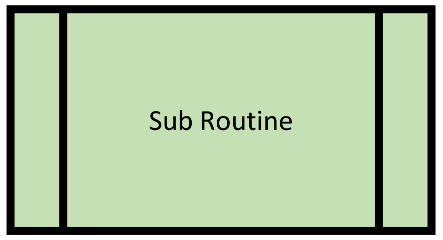
This shape represents the start or end of the process.



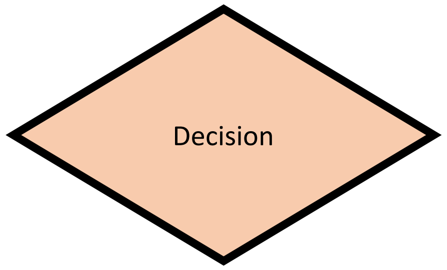
This shape represents the input or output of data.



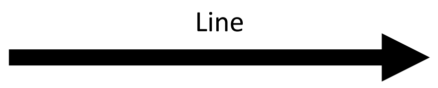
This shape represents something being initialised, processed or calculated.



This shape represents a subroutine call that will relate to a separate non-linked flow chart.



This shape represents a decision with yes or no, true or false that results in two lines for the two outcomes.



An arrow represents control passing between connected shapes.

J277 - 2.1 Algorithms

Pseudocode

Pseudocode uses short English words/statements to describe an algorithm.

It would generally look a little more structured than just writing English sentences.

However it is very flexible.

It is less precise than using a reference language, or a programming language.

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

J277 - 2.1 Algorithms

Exam reference language

Output:

```
print("Hello")
```

Input:

```
num = input("Enter a number")
```

Selection:

```
if num == 2 then
        ...
elseif num < 4 then
        ...
endif
```

FOR Loops

```
for i = 1 to 10
        ...
next i
```

WHILE Loops

```
while (i != 11)
        ...
endwhile
```

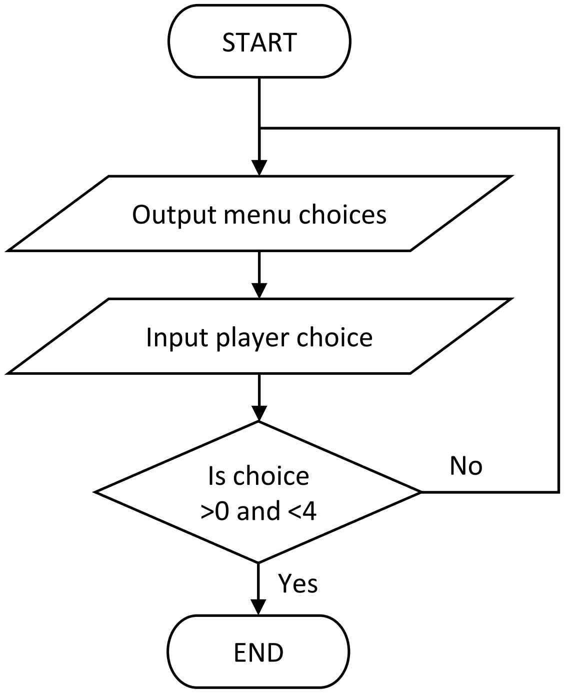
```
do
        ...
until i > 10
```

How to produce algorithms using flow diagrams

An algorithm for an RPG game displays 3 choices from a menu and allows the user to enter their choice.

1. Play game
2. Change character
3. Quit

The user input is validated so only the numbers 1-3 can be entered.



J277 - 2.1 Algorithms

Interpret, correct, refine or complete algorithms.

An algorithm for an RPG game displays 3 choices from a menu and allows the user to enter their choice.

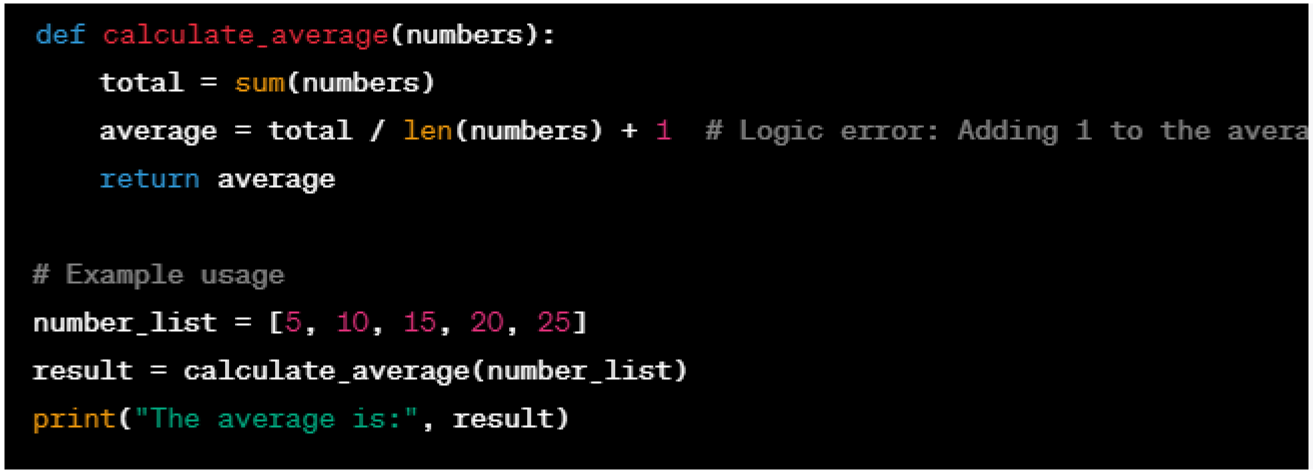
1. Play game
2. Change character
3. Quit

The user input is validated so only the numbers 1-3 can be entered.

```
do
    print(“1. Play game”)
    print(“2. Change character”)
    print(“3. Quit”)

    input(int(choice))

until choice<1 OR choice>3
```

J277 - 2.1 Algorithms

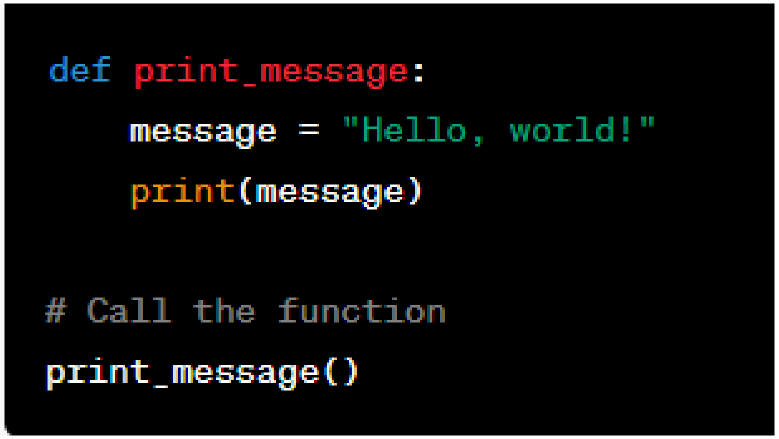
Identifying common errors and suggesting fixes



```
def calculate_average(numbers):
    total = sum(numbers)
    average = total / len(numbers) + 1  # Logic error: Adding 1 to the avera
    return average

# Example usage
number_list = [5, 10, 15, 20, 25]
result = calculate_average(number_list)
print("The average is:", result)
```

The error is on:	`average = total / len(numbers) + 1`
The type of error is:	Logical
In order to fix this error:	Instead of calculating the correct average, the program mistakenly adds 1 to the average value. `average = total / len(numbers)`

Identifying common errors and suggesting fixes



```
def print_message:
    message = "Hello, world!"
    print(message)

# Call the function
print_message()
```

The error is on:	`Def print_message`
The type of error is:	Syntax
In order to fix this error:	Includes the required parentheses after the function name. `def print_message():`

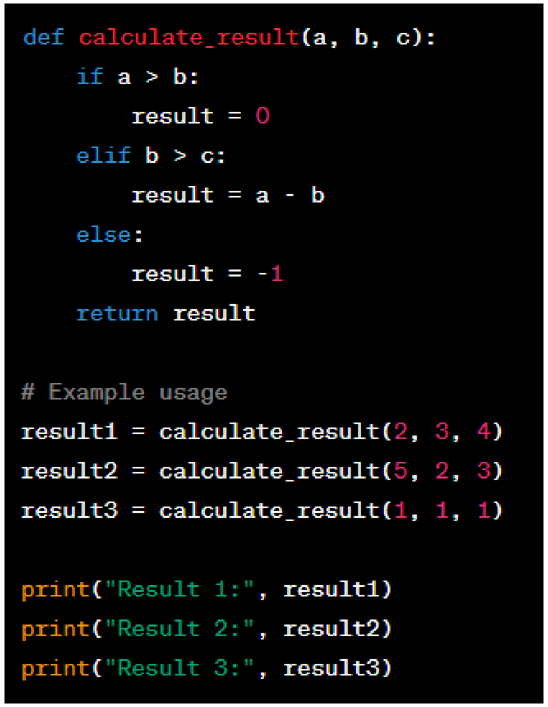
J277 - 2.1 Algorithms

Trace tables

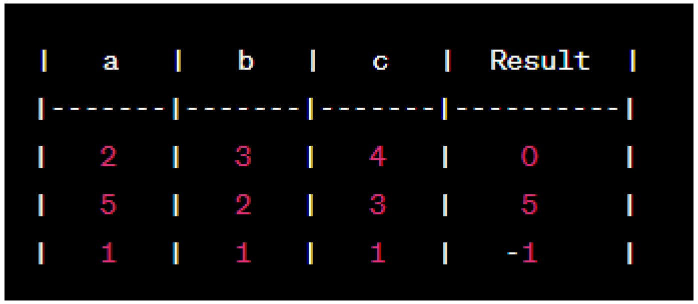
In this example, the trace table represents the input values **a**, **b**, and **c**, as well as the expected result for each combination. The Python code defines a function called **calculate_result** that takes three parameters: **a**, **b**, and **c**.

The logic in the code checks different conditions using **if**, **elif**, and **else** statements to determine the appropriate result based on the given inputs. The function then returns the calculated result.

The example usage section calls the **calculate_result** function with different sets of input values, and the results are stored in **result1**, **result2**, and **result3**. Finally, the program prints out the calculated results.



```python
def calculate_result(a, b, c):
    if a > b:
        result = 0
    elif b > c:
        result = a - b
    else:
        result = -1
    return result

# Example usage
result1 = calculate_result(2, 3, 4)
result2 = calculate_result(5, 2, 3)
result3 = calculate_result(1, 1, 1)

print("Result 1:", result1)
print("Result 2:", result2)
print("Result 3:", result3)
```



a	b	c	Result
2	3	4	0
5	2	3	5
1	1	1	-1

J277 - 2.1 Algorithms

2.1.3 Searching and sorting algorithms

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

Required

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

Not required

- ✗ To remember the code for these algorithms

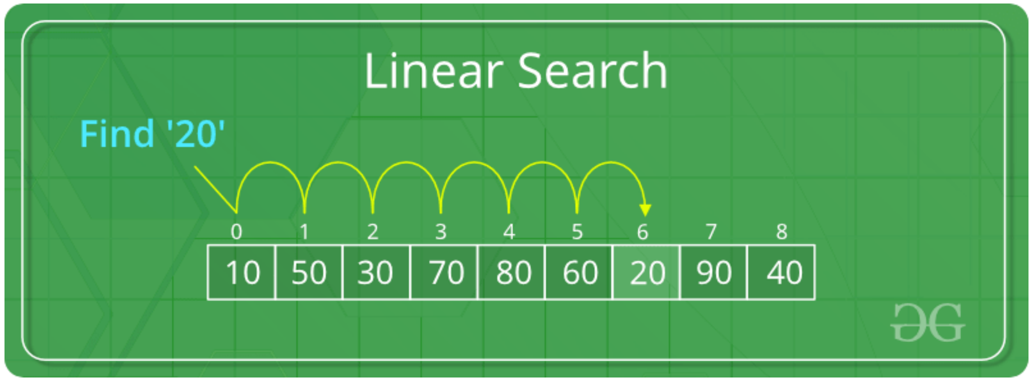
J277 - 2.1 Algorithms

Linear search

Explanation of a linear search:

Each item in the list is checked in order. Only works on an ordered list.

- Check the first value
- IF it is the value you are looking for
    - Celebrate and stop
- ELSE move to and check the next value
- REPEAT UNTIL you have checked all the elements and not found the value you are looking for

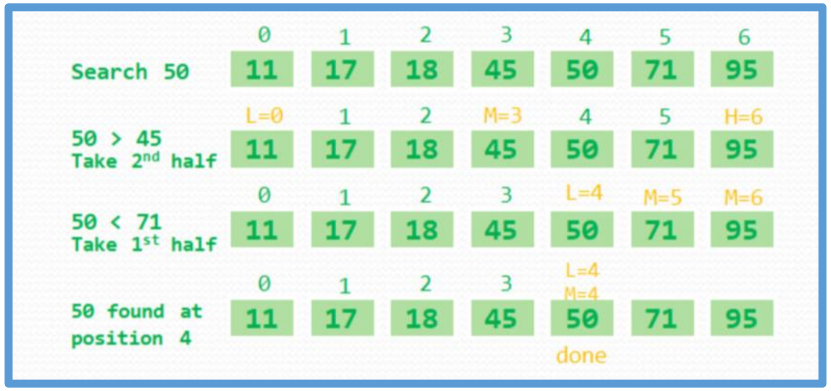


J277 - 2.1 Algorithms

Binary search

Explanation of a binary search:

Calculate the mid point. Check if that is the item to find. If not, if it is lower than the midpoint, repeat on the left half of the list, or repeat on the right half of the list.



The list needs to be in order.
Take the middle value.
Compare to the value you are looking for.
IF it is the value you are looking for.
- Celebrate, and stop.

ELSEIF it is larger than the one you are looking for.
- Take the values to the **left** of the middle value.

IF it is smaller than the one you are looking for.
- Take the values to the **right** of the middle value.

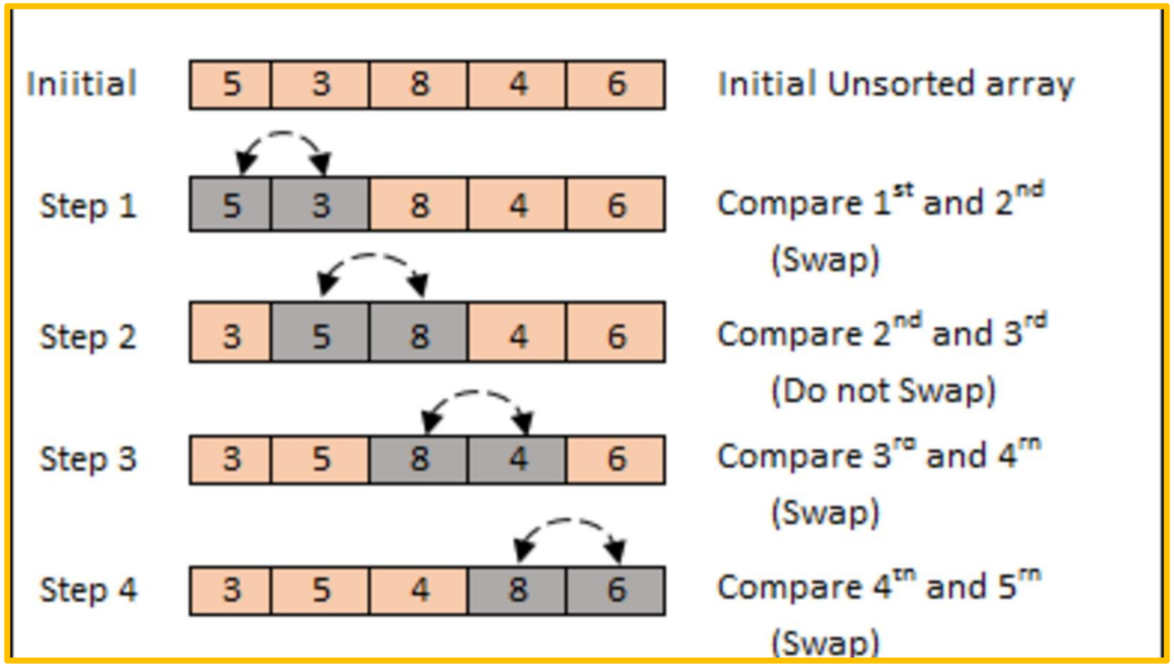
Repeat with the new list.

J277 - 2.1 Algorithms

Bubble sort

Moving through a list repeatedly, swapping elements that are in the wrong order.

```
1. Take the first element and second element from the list
2. Compare them
3. IF element 1 > element 2 THEN
       – Swap then
4. ELSE
       – Do nothing
5. Repeat: Move along the list to the next pair
       – IF no more elements: Goto 1
       – ELSE: Goto 2
   Until: you have moved through the entire list and not made any changes
```



Iniitial
5 3 8 4 6
Initial Unsorted array
Step 1
5 3 8 4 6
Compare 1st and 2nd (Swap)
Step 2
3 5 8 4 6
Compare 2nd and 3rd (Do not Swap)
Step 3
3 5 8 4 6
Compare 3rd and 4th (Swap)
Step 4
3 5 4 8 6
Compare 4th and 5th (Swap)

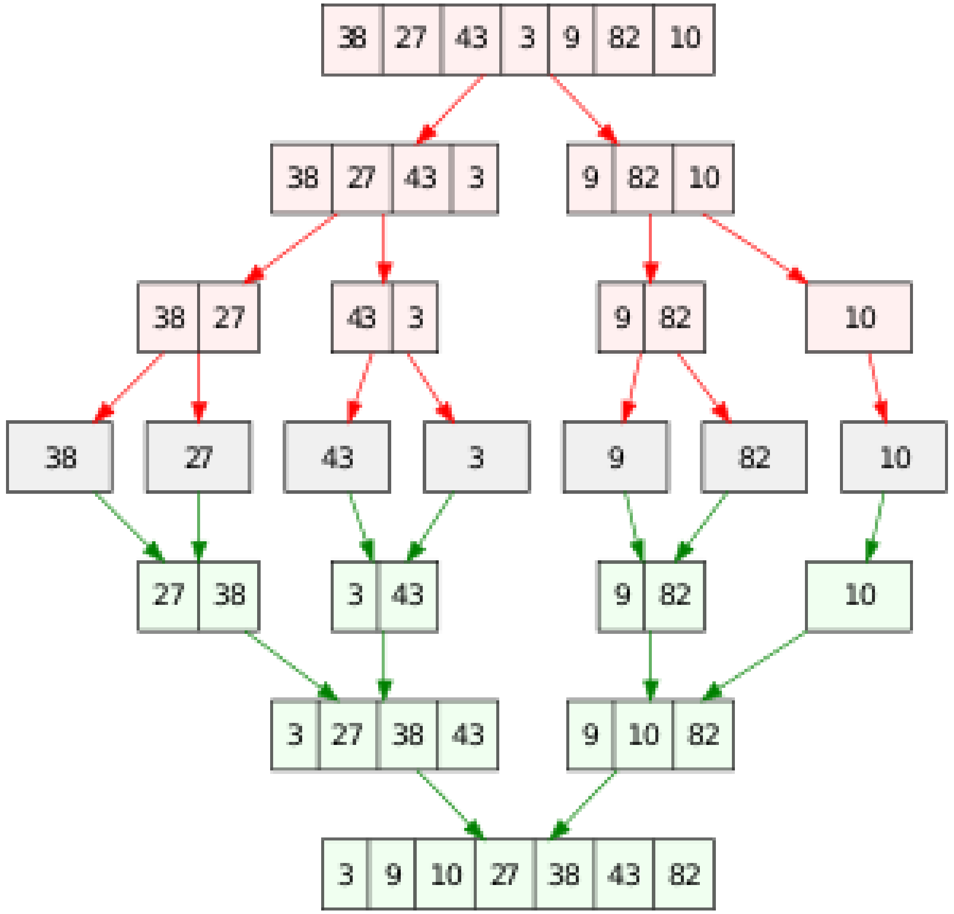
Continue until there are no more swaps.

J277 - 2.1 Algorithms

Merge sort

A list is split into individual lists, these are then combined (2 lists at a time).

1. Split all elements into individual lists.
2. Compare the first element in both lists.
3. Put the smallest into a new list.
4. Compare the next element of 1 list with the second element of the 2nd list.
5. Put the smallest into a new list.
6. Repeat until merged.

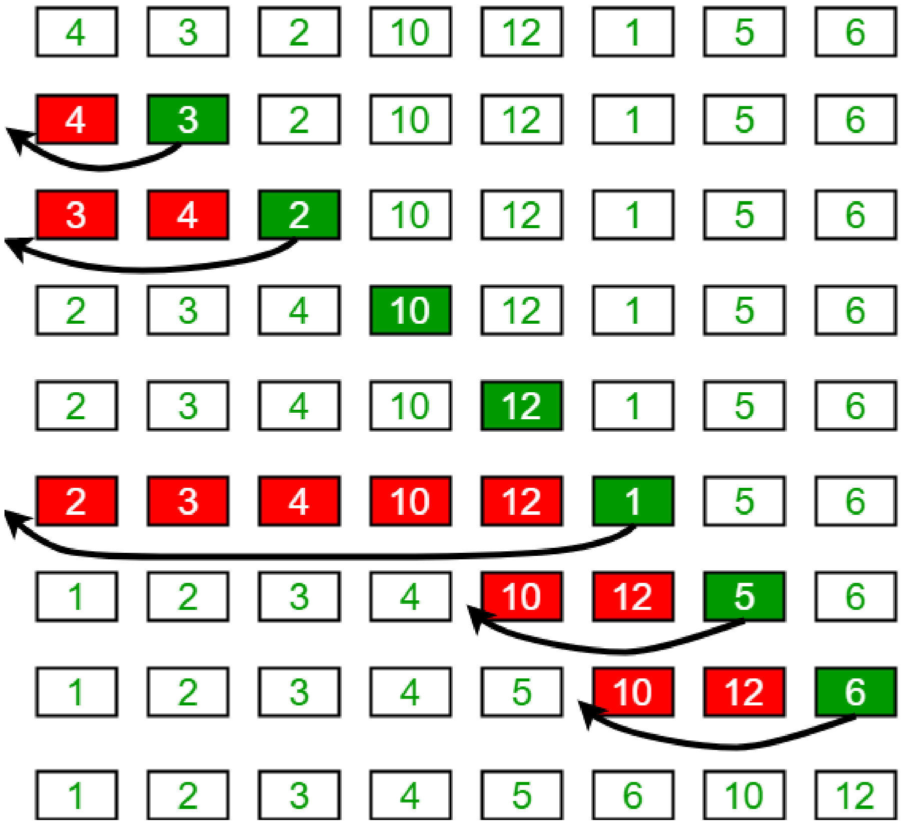


J277 - 2.1 Algorithms

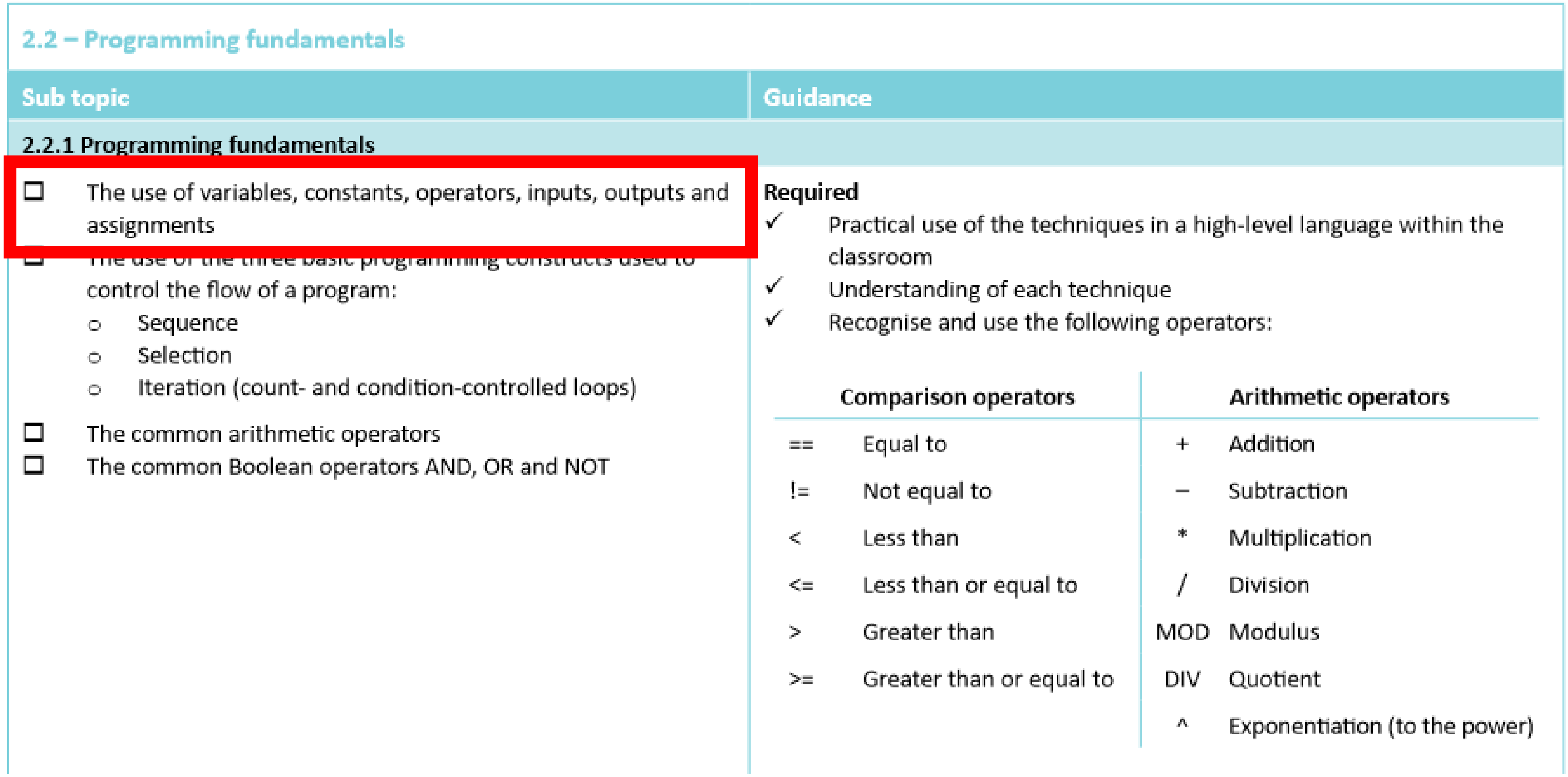
Insertion sort

Each items is take in turn, compare to the items in a sorted list and placed in the correct position.

1. Element 1 is a 'sorted' list.
2. The rest of the elements are an 'unsorted' list.
3. Compare the first element in the 'unsorted' list to each element in the sorted list.
4. IF it is smaller, put it in in front of that element (move the others along).
5. ELSEIF it is larger, compare with the next.
6. ELSEIF there are no more elements in the 'sorted' list put it in the final position.
7. REPEAT UNTIL all element in the 'unsorted' list are in the 'sorted' list.



J277 - 2.2 Programming fundamentals



2.2 – Programming fundamentals

Sub topic	Guidance
2.2.1 Programming fundamentals	
❑ The use of variables, constants, operators, inputs, outputs and assignments ❑ The use of the three basic programming constructs used to control the flow of a program: ○ Sequence ○ Selection ○ Iteration (count- and condition-controlled loops) ❑ The common arithmetic operators ❑ The common Boolean operators AND, OR and NOT	**Required** ✓ Practical use of the techniques in a high-level language within the classroom ✓ Understanding of each technique ✓ Recognise and use the following operators:

Comparison operators		Arithmetic operators	
==	Equal to	+	Addition
!=	Not equal to	–	Subtraction
<	Less than	*	Multiplication
<=	Less than or equal to	/	Division
>	Greater than	MOD	Modulus
>=	Greater than or equal to	DIV	Quotient
		^	Exponentiation (to the power)

J277 - 2.2 Programming fundamentals

Variables, constants, inputs, outputs and assignments

Variable:	A value stored in memory that can change whilst the program is running.
Constant:	A value that does not change whilst the program is running and is assigned when the program is designed.
Assignment:	Giving a variable or constant a value.

constant

input

Example annotated program: Python

```
#Program to calculate the VAT of an ex-VAT item.

vat_rate = 0.2

cost = input("Enter the ex-VAT price of the item: £")

cost = float(cost)

vat = round(cost * vat_rate,2)

total = cost + vat

print("Ex-VAT price was: £","{:.2f}".format(cost))
print("VAT is: £","{:.2f}".format(vat))
print("Total price is: £","{:.2f}".format(total))
```

variable

assignment

output

constant

input

Example annotated program: OCR reference language

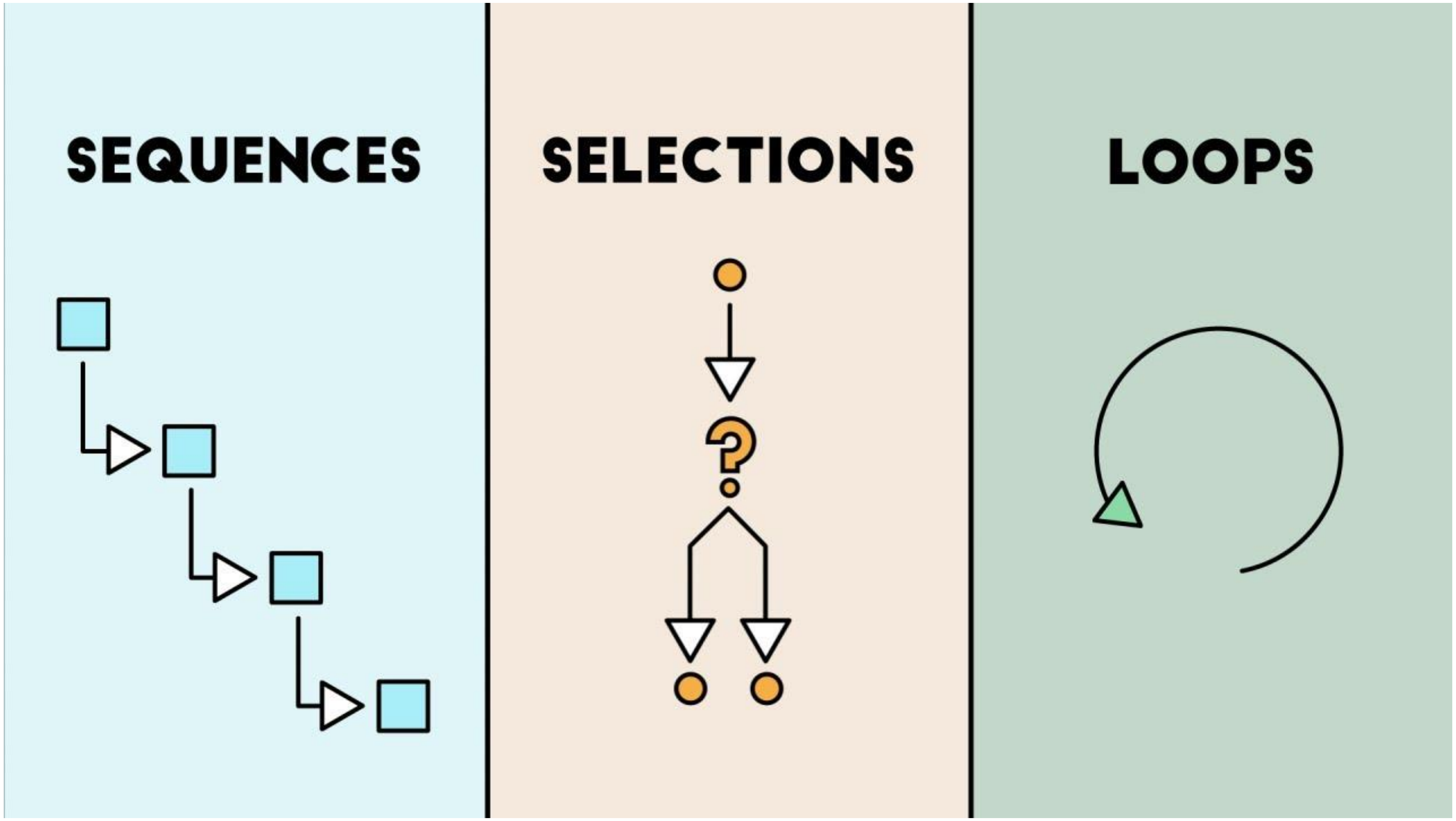
```
//Program to calculate the VAT of an ex-VAT item.

const vat_rate = 0.2

cost = input("Enter the ex-VAT price of the item: £")

cost = float(cost)

vat = round(cost * vat_rate,2)

total = cost + vat

print("Ex-VAT price was: £", (cost))
print("VAT is: £", (vat))
print("Total price is: £", (total))
```

variable

assignment

output

The use of the three basic programming constructs: sequence, selection and iteration



J277 - 2.2 Programming fundamentals

The use of the three basic programming constructs: sequence, selection and iteration

Sequence

```
x = 3
MyVariable = 34
New_Variable = x + MyVariable
```

Iteration with a count controlled loop

```
for i in range(10):
  print("Mrs Rollings")
next counter
```

Selection

```
if x > 90 then
        output "a*"
else if x > 80 then
   output "a"
else if x > 70 then
        output "b"
else
        output "fail"
end if
```

Iteration with a condition controlled loop

```
answer = "y"
while answer == "y":
   print("Stay very still")
   answer = input("Is the monster friendly? y/n")
print('Run away!')
end while
```

J277 - 2.2 Programming fundamentals

The use of the three basic programming constructs: sequence, selection and iteration

Selection

```
select case x
    case >90
           output “a*”
    case >80
           output “a”
    case >70
           output “b”
…
    case else
           output “fail”
end select
```

Not supported by all languages, this construct is an alternative to using else if or elif commands.

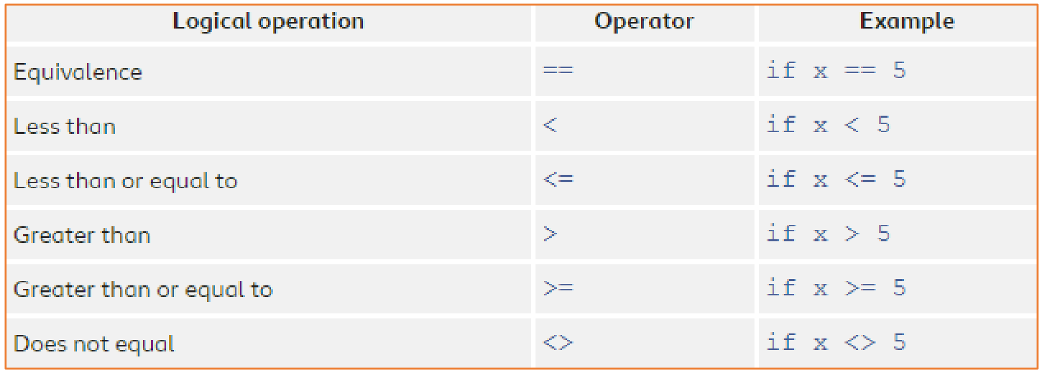
Iteration with a condition controlled loop

```
do
    name = input("Enter a chosen name: ")
until name.length < 4 or name.length > 12
```

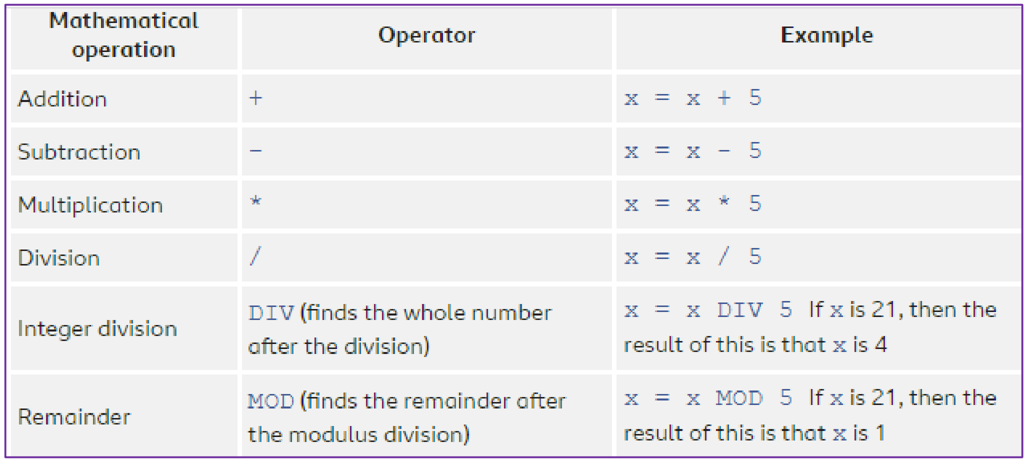
Not supported by all languages, this iteration is different because it will execute the code inside the loop at least once.

J277 - 2.2 Programming fundamentals

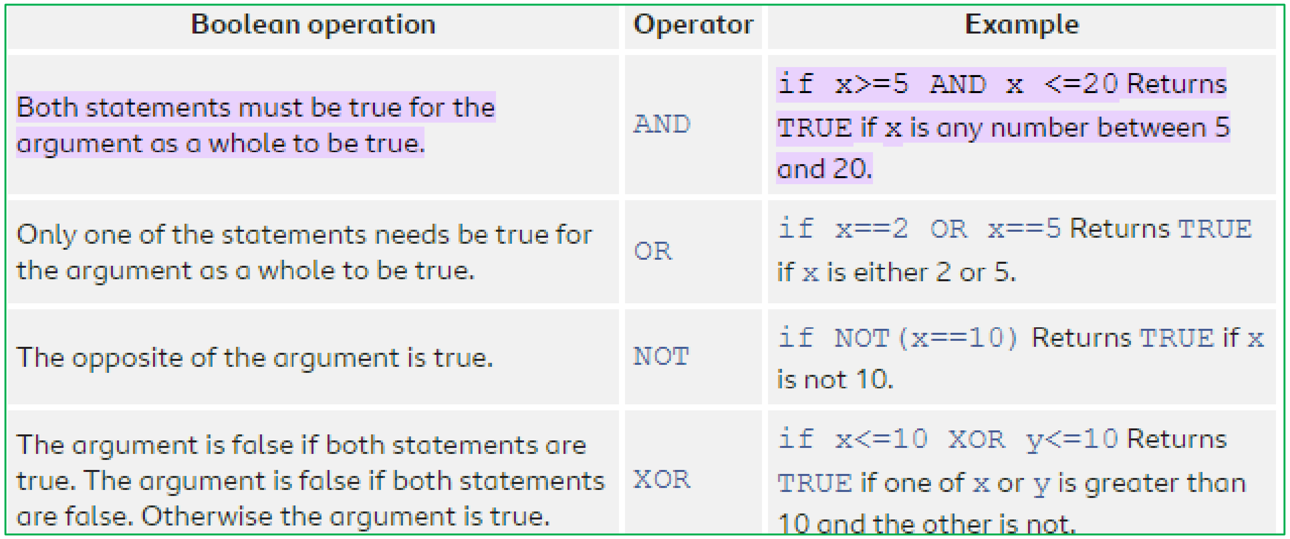
Arithmetic, comparison and Boolean operators



Logical operation	Operator	Example
Equivalence	`==`	`if x == 5`
Less than	`<`	`if x < 5`
Less than or equal to	`<=`	`if x <= 5`
Greater than	`>`	`if x > 5`
Greater than or equal to	`>=`	`if x >= 5`
Does not equal	`<>`	`if x <> 5`

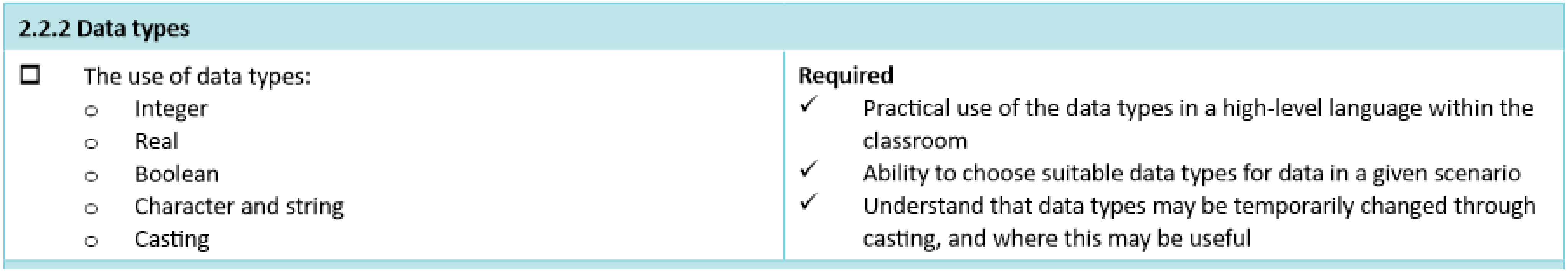


Mathematical operation	Operator	Example
Addition	`+`	`x = x + 5`
Subtraction	`-`	`x = x - 5`
Multiplication	`*`	`x = x * 5`
Division	`/`	`x = x / 5`
Integer division	`DIV` (finds the whole number after the division)	`x = x DIV 5` If `x` is 21, then the result of this is that `x` is 4
Remainder	`MOD` (finds the remainder after the modulus division)	`x = x MOD 5` If `x` is 21, then the result of this is that `x` is 1



Boolean operation	Operator	Example
Both statements must be true for the argument as a whole to be true.	`AND`	`if x>=5 AND x <=20` Returns `TRUE` if `x` is any number between 5 and 20.
Only one of the statements needs be true for the argument as a whole to be true.	`OR`	`if x==2 OR x==5` Returns `TRUE` if `x` is either 2 or 5.
The opposite of the argument is true.	`NOT`	`if NOT(x==10)` Returns `TRUE` if `x` is not 10.
The argument is false if both statements are true. The argument is false if both statements are false. Otherwise the argument is true.	`XOR`	`if x<=10 XOR y<=10` Returns `TRUE` if one of `x` or `y` is greater than 10 and the other is not.

J277 - 2.2 Programming fundamentals

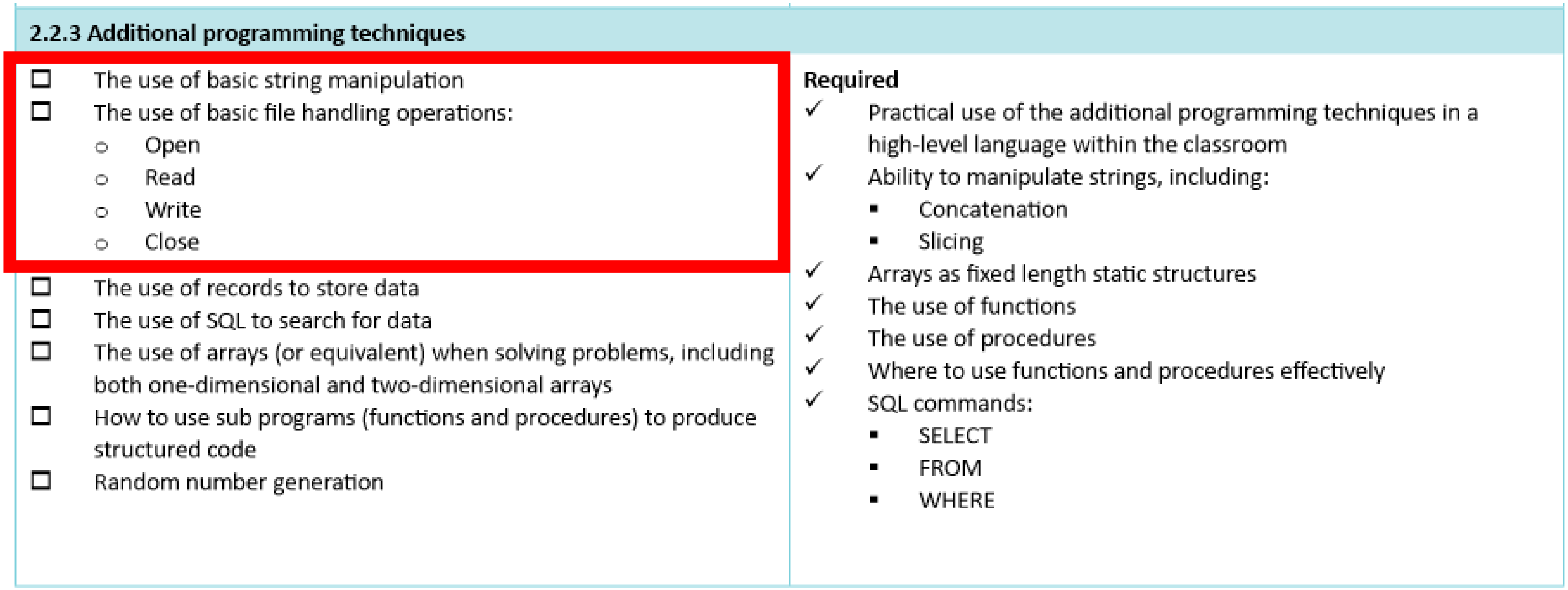


2.2.2 Data types	
☐ The use of data types: ○ Integer ○ Real ○ Boolean ○ Character and string ○ Casting	**Required** ✓ Practical use of the data types in a high-level language within the classroom ✓ Ability to choose suitable data types for data in a given scenario ✓ Understand that data types may be temporarily changed through casting, and where this may be useful

J277 - 2.2 Programming fundamentals
Data types

Data Type	Definition	Examples
Integer	A whole number.	1, 5, -63, 247
Float	A number with a decimal point	1.5, 3.14159, -0.47, 2.0
String	A block of text (designated with ‘ marks in Python). Any numbers contained within a string are treated as text and cannot be used in calculations	‘Python’, ‘ICT’, ‘Hilbre High’, ‘3’
Array (called List in Python)	A list of data stored in a structured way. Represented with [ ] in Python and values separated with a comma.	[‘Python’, ‘programming’, ‘is’, ‘awesome’]
Comment	A comment in the code which is ignored by the computer. Useful for leaving instructions	# This line creates a variable called myBox

J277 - 2.2 Programming fundamentals



2.2.3 Additional programming techniques

- ☐ The use of basic string manipulation
- ☐ The use of basic file handling operations:
  - ○ Open
  - ○ Read
  - ○ Write
  - ○ Close
- ☐ The use of records to store data
- ☐ The use of SQL to search for data
- ☐ The use of arrays (or equivalent) when solving problems, including both one-dimensional and two-dimensional arrays
- ☐ How to use sub programs (functions and procedures) to produce structured code
- ☐ Random number generation

Required

- ✓ Practical use of the additional programming techniques in a high-level language within the classroom
- ✓ Ability to manipulate strings, including:
  - ▪ Concatenation
  - ▪ Slicing
- ✓ Arrays as fixed length static structures
- ✓ The use of functions
- ✓ The use of procedures
- ✓ Where to use functions and procedures effectively
- ✓ SQL commands:
  - ▪ SELECT
  - ▪ FROM
  - ▪ WHERE

J277 - 2.2 Programming fundamentals

Casting operations

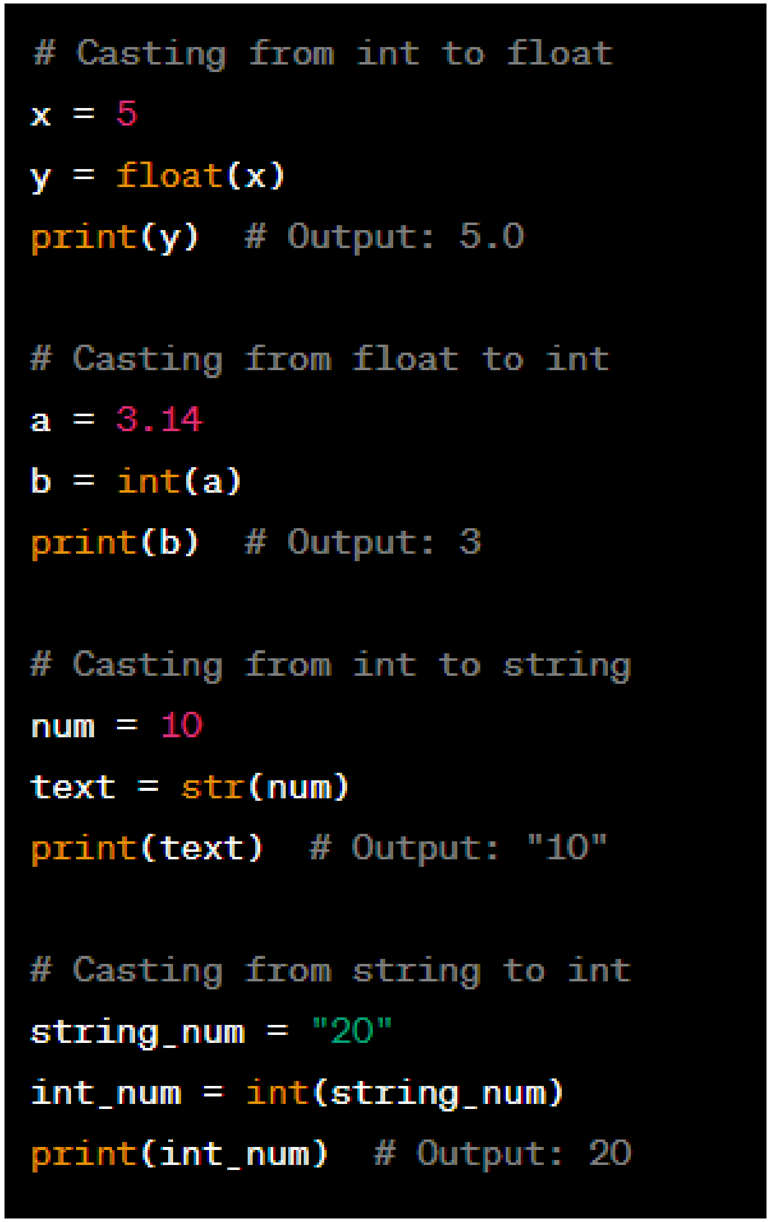
Casting:

A process that converts a variable's data type into another data type.

Examples

cost = int(cost)

cost = str(cost)



```
# Casting from int to float
x = 5
y = float(x)
print(y)  # Output: 5.0

# Casting from float to int
a = 3.14
b = int(a)
print(b)  # Output: 3

# Casting from int to string
num = 10
text = str(num)
print(text)  # Output: "10"

# Casting from string to int
string_num = "20"
int_num = int(string_num)
print(int_num)  # Output: 20
```

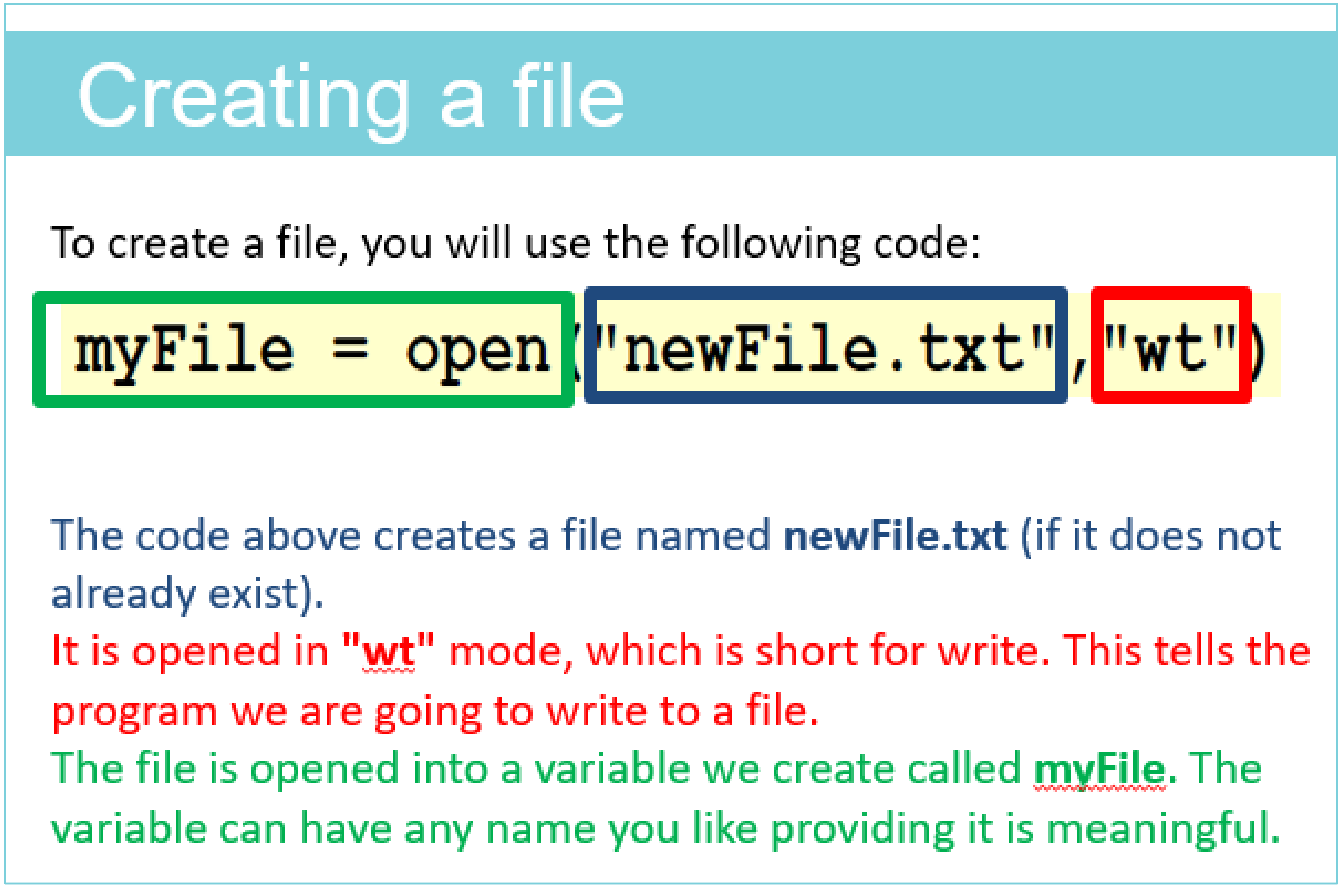
J277 - 2.2 Programming fundamentals

Basic string manipulation

Assuming that a string called name is assigned the value: “Hilbre12345”:

To return the **length** of a string:	l = len(name)	OCR Reference Language: l = name.length
To return the string in **uppercase**:	upperC = name.upper()	OCR Reference Language: up = name.upper
To return the string in **lowercase**:	lowerC = name.lower()	OCR Reference Language: low = name.upper
To return the 5th character of the string:	char = name[4:5]	OCR Reference Language: char = name.substring(4,1)

Basic file handling operations: Opening a file



Creating a file

To create a file, you will use the following code:

```
myFile = open("newFile.txt","wt")
```

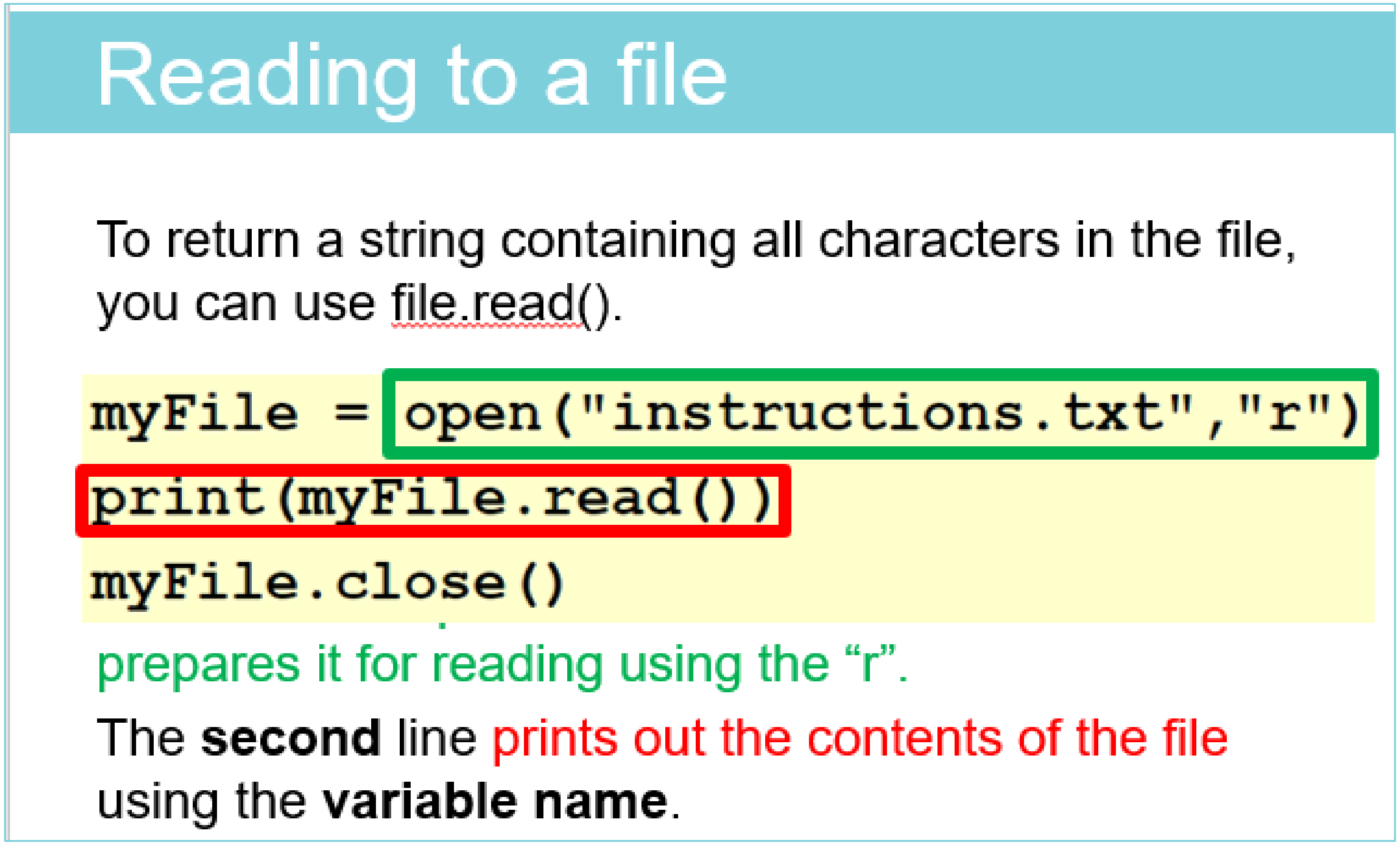
The code above creates a file named **newFile.txt** (if it does not already exist).

It is opened in "**wt**" mode, which is short for write. This tells the program we are going to write to a file.

The file is opened into a variable we create called **myFile**. The variable can have any name you like providing it is meaningful.

J277 - 2.2 Programming fundamentals

Basic file handling operations: Reading a file



Reading to a file

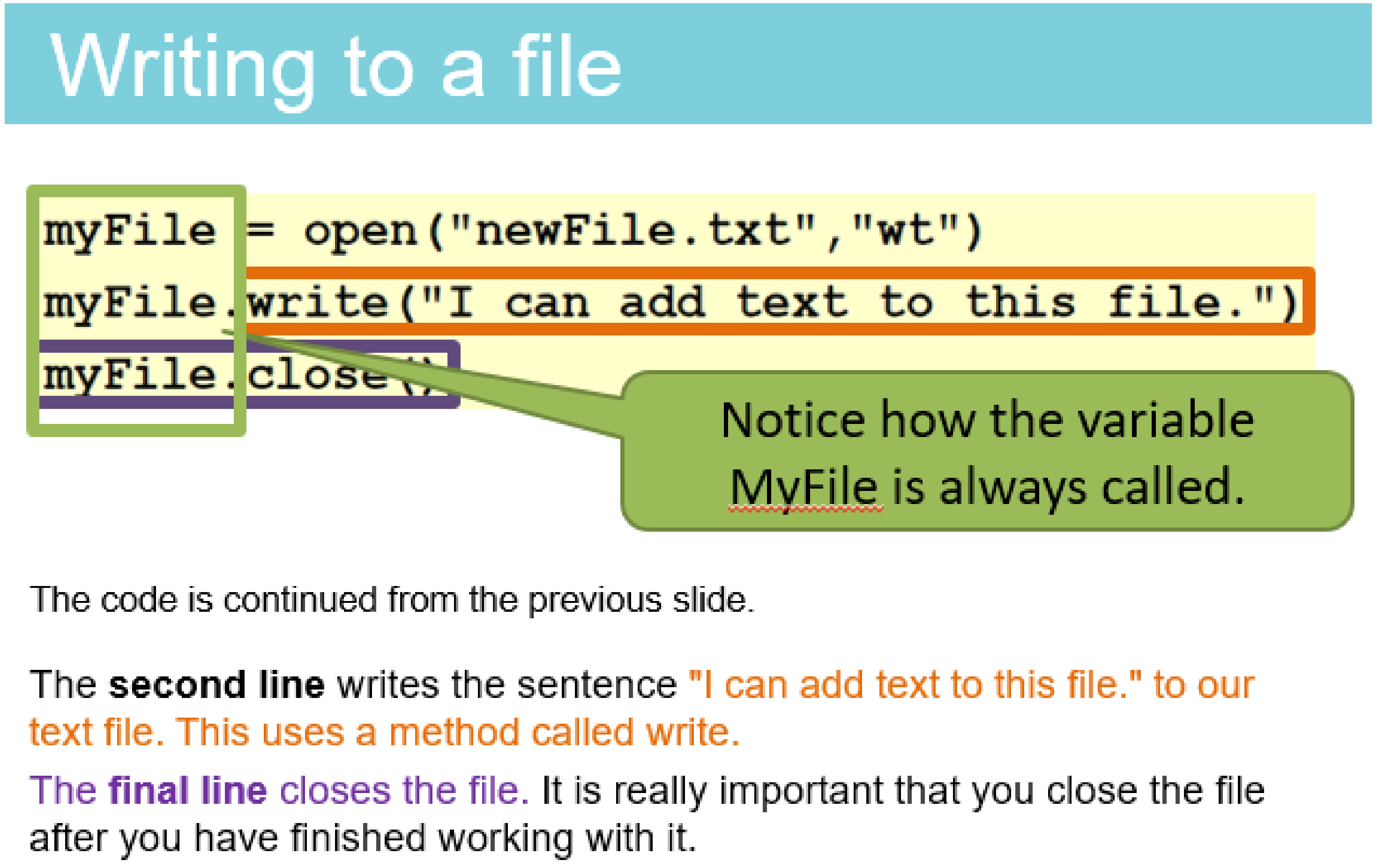
To return a string containing all characters in the file, you can use file.read().

```
myFile = open("instructions.txt","r")
print(myFile.read())
myFile.close()
```

prepares it for reading using the “r”.

The **second** line prints out the contents of the file using the **variable name**.

Basic file handling operations: Writing to a file



Writing to a file

```
myFile = open("newFile.txt","wt")
myFile.write("I can add text to this file.")
myFile.close()
```

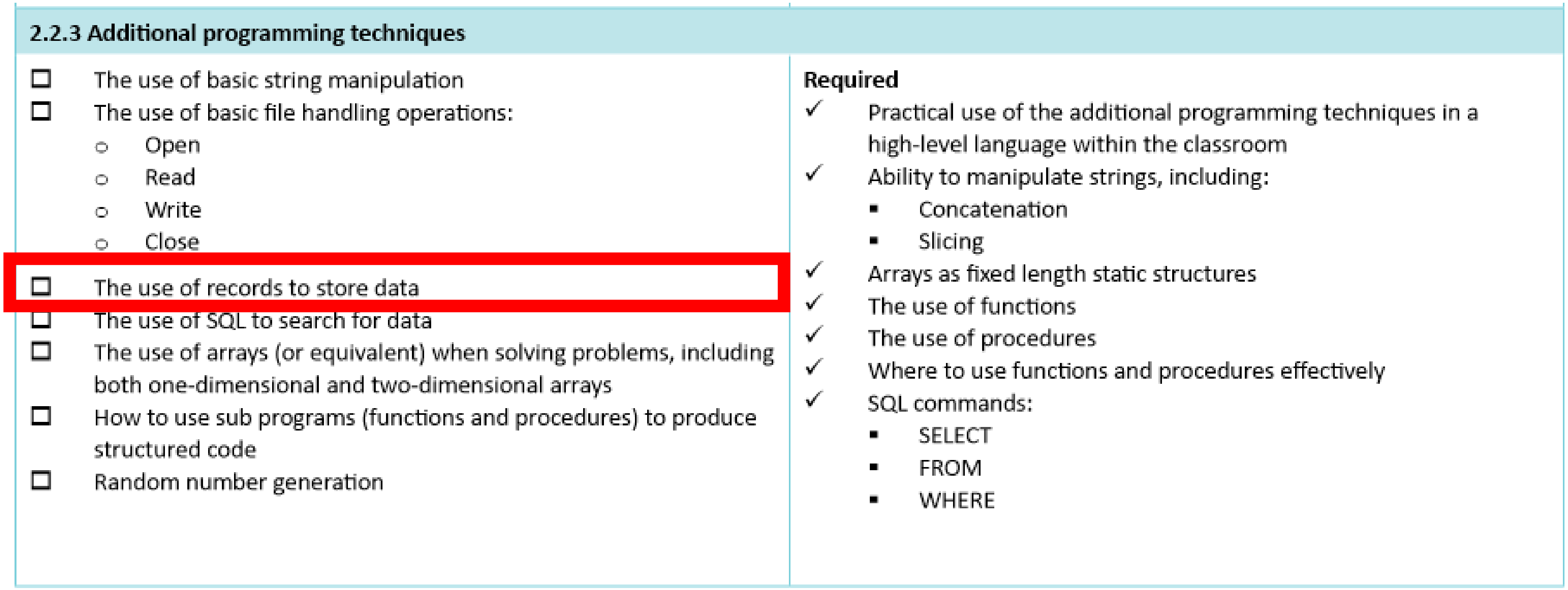
Notice how the variable MyFile is always called.

The code is continued from the previous slide.

The **second line** writes the sentence "I can add text to this file." to our text file. This uses a method called write.

The **final line** closes the file. It is really important that you close the file after you have finished working with it.

J277 - 2.2 Programming fundamentals



2.2.3 Additional programming techniques	
☐ The use of basic string manipulation ☐ The use of basic file handling operations: ○ Open ○ Read ○ Write ○ Close ☐ The use of records to store data ☐ The use of SQL to search for data ☐ The use of arrays (or equivalent) when solving problems, including both one-dimensional and two-dimensional arrays ☐ How to use sub programs (functions and procedures) to produce structured code ☐ Random number generation	**Required** ✓ Practical use of the additional programming techniques in a high-level language within the classroom ✓ Ability to manipulate strings, including: ▪ Concatenation ▪ Slicing ✓ Arrays as fixed length static structures ✓ The use of functions ✓ The use of procedures ✓ Where to use functions and procedures effectively ✓ SQL commands: ▪ SELECT ▪ FROM ▪ WHERE

J277 - 2.2 Programming fundamentals

Use of records

Record: A data structure that stores related values of different data types.

Field: An element of a record used to store one piece of data.

Most languages let you define your own data structures in the form of records.

An example:

Create a fixed record structure by giving a data type and name for each filed.

```
Record ticket

      string filmName
      int seatNumber
      real price
endrecord
```

Create records by giving values for each field

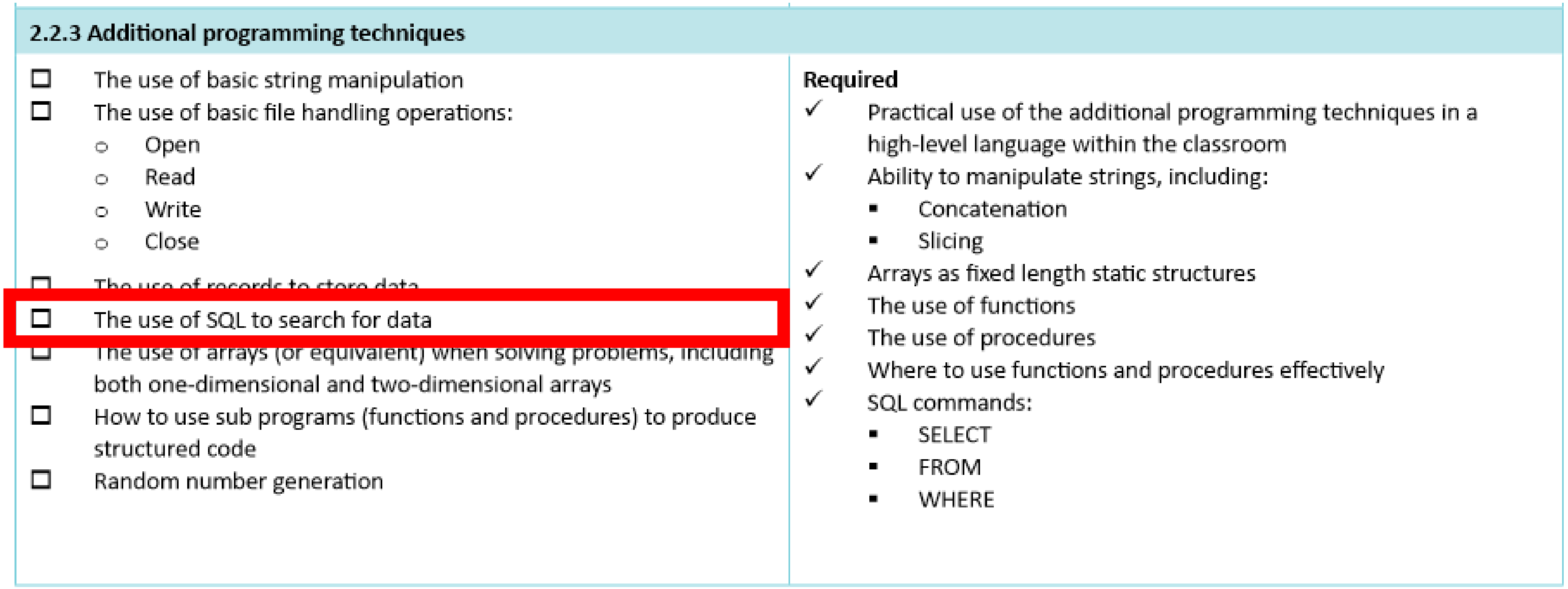
```
Ticket1 = ticket (“Data Force“, 16, 7.50)
```

Use the variable and field name to access the values

```
Print (ticket1.seatNumber)
```

```
16
```

J277 - 2.2 Programming fundamentals



2.2.3 Additional programming techniques	
☐ The use of basic string manipulation ☐ The use of basic file handling operations: ○ Open ○ Read ○ Write ○ Close ☐ The use of records to store data ☐ The use of SQL to search for data ☐ The use of arrays (or equivalent) when solving problems, including both one-dimensional and two-dimensional arrays ☐ How to use sub programs (functions and procedures) to produce structured code ☐ Random number generation	**Required** ✓ Practical use of the additional programming techniques in a high-level language within the classroom ✓ Ability to manipulate strings, including: ▪ Concatenation ▪ Slicing ✓ Arrays as fixed length static structures ✓ The use of functions ✓ The use of procedures ✓ Where to use functions and procedures effectively ✓ SQL commands: ▪ SELECT ▪ FROM ▪ WHERE

J277 - 2.2 Programming fundamentals

The use of records to store data & SQL to search for data

Structured query language (SQL):

SQL is a programming language used for interrogating a database.

Person ID	Title	Forename	Surname	Email address
1001	Mr	Alan	Turing	aturing@bitesize.com
1002	Mrs	Ada	Lovelace	alovelace@gcsecompsci.com
1003	Miss	Grace	Hopper	ghopper@bitesizemail.co.uk
1004	Mr	George	Boole	gboole@bbcbitesize.com

Retrieving data

Data can be retrieved using the **commands** `SELECT`, `FROM` and `WHERE`, for example:

```
SELECT * FROM "personnel" WHERE "Title" = "Mr"
```

Note - `*` stands for wildcard, which means all records. This would retrieve the following data:

1001 Mr Alan Turing aturing@bitesize.com

1004 Mr George Boole gboole@bbcbitesize.com

The `LIKE` command can be used to find matches for an incomplete word, for example:

```
SELECT * FROM "personnel" WHERE "email address" LIKE "%com"
```

This would retrieve:

1001 Mr Alan Turing aturing@bitesize.com

1002 Mrs Ada Lovelace alovelace@gcsecompsci.com

1004 Mr George Boole gboole@bbcbitesize.com

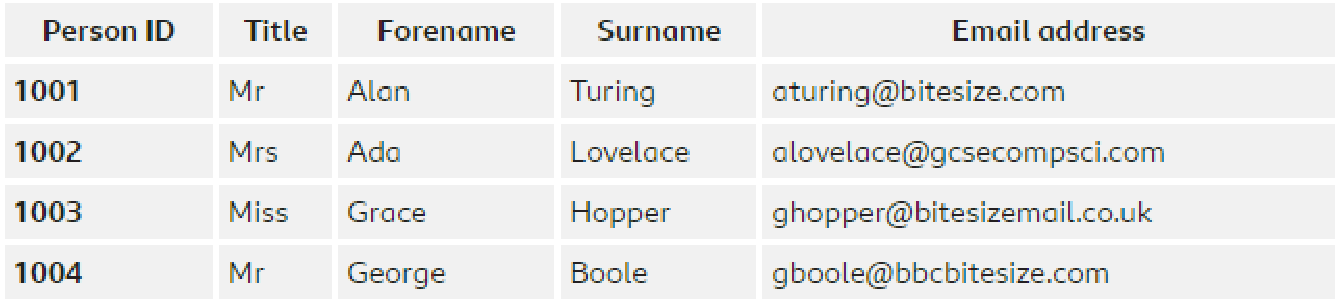
Note - `%com` is also a wildcard which will return any value that contains “com”.

J277 - 2.2 Programming fundamentals

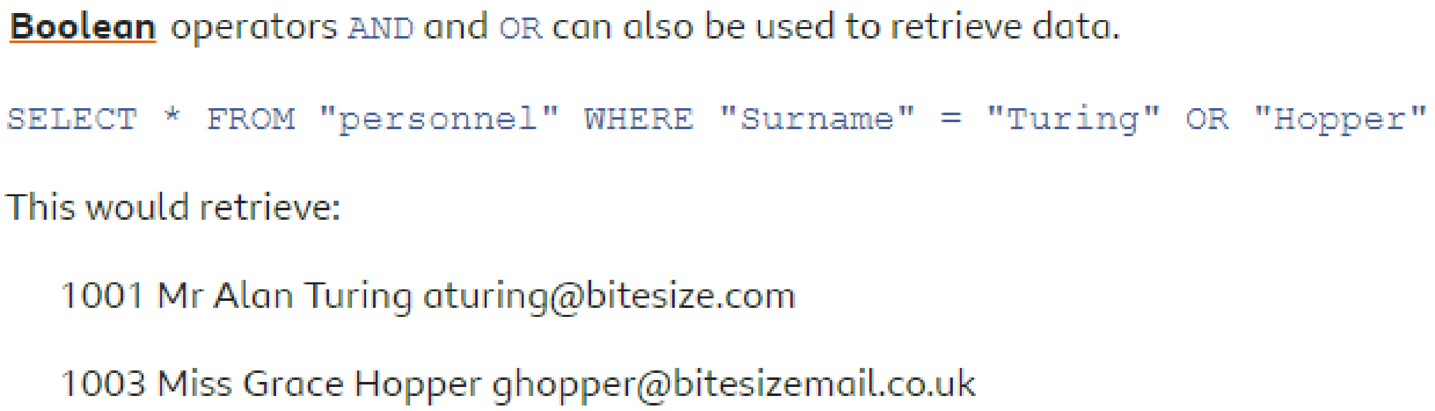
The use of records to store data & SQL to search for data

Structured query language (SQL):

SQL is a programming language used for interrogating a database.



Person ID	Title	Forename	Surname	Email address
1001	Mr	Alan	Turing	aturing@bitesize.com
1002	Mrs	Ada	Lovelace	alovelace@gcsecompsci.com
1003	Miss	Grace	Hopper	ghopper@bitesizemail.co.uk
1004	Mr	George	Boole	gboole@bbcbitesize.com



Boolean operators `AND` and `OR` can also be used to retrieve data.

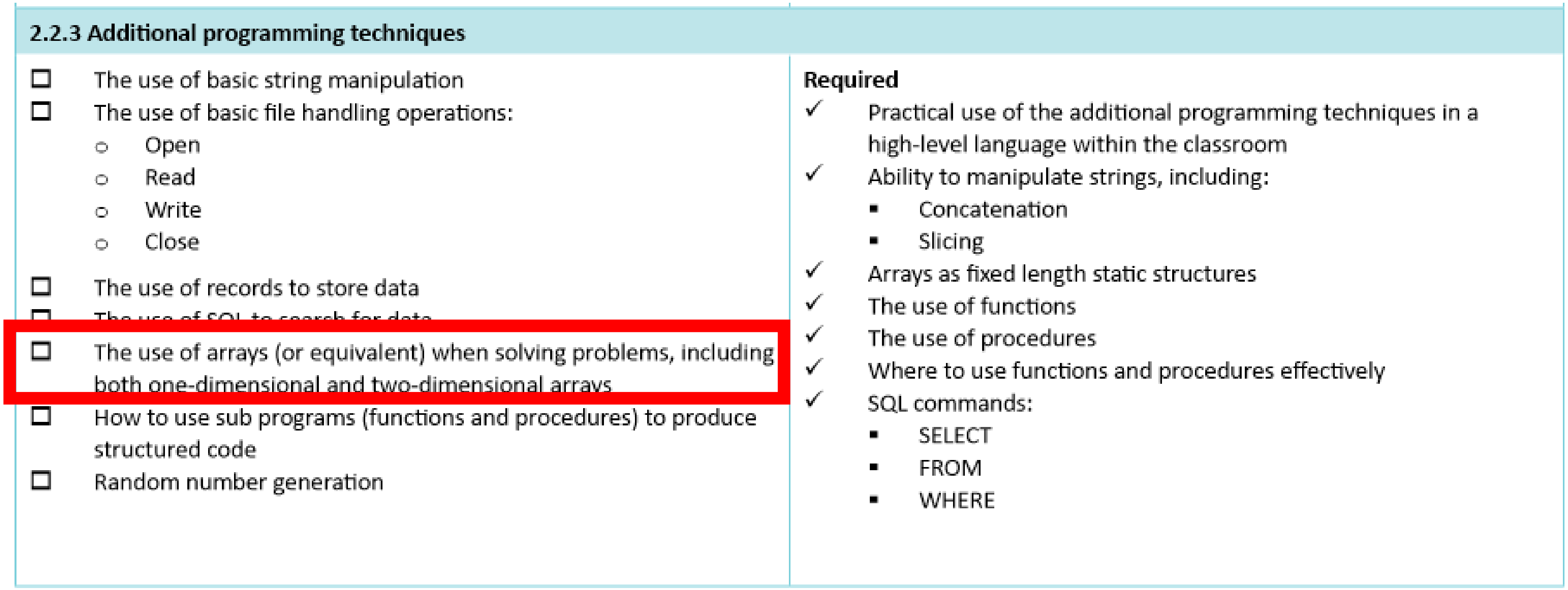
```
SELECT * FROM "personnel" WHERE "Surname" = "Turing" OR "Hopper"
```

This would retrieve:

1001 Mr Alan Turing aturing@bitesize.com

1003 Miss Grace Hopper ghopper@bitesizemail.co.uk

J277 - 2.2 Programming fundamentals



2.2.3 Additional programming techniques

- ☐ The use of basic string manipulation
- ☐ The use of basic file handling operations:
  - Open
  - Read
  - Write
  - Close
- ☐ The use of records to store data
- ☐ The use of SQL to search for data
- ☐ The use of arrays (or equivalent) when solving problems, including both one-dimensional and two-dimensional arrays
- ☐ How to use sub programs (functions and procedures) to produce structured code
- ☐ Random number generation

Required

- ✓ Practical use of the additional programming techniques in a high-level language within the classroom
- ✓ Ability to manipulate strings, including:
  - Concatenation
  - Slicing
- ✓ Arrays as fixed length static structures
- ✓ The use of functions
- ✓ The use of procedures
- ✓ Where to use functions and procedures effectively
- ✓ SQL commands:
  - SELECT
  - FROM
  - WHERE

J277 - 2.2 Programming fundamentals

Arrays

An array is a series of memory locations – or ‘boxes’ – each of which holds a single item of data, but with each box sharing the same name. All data in an array must be of the same **data type**.

For example, imagine that a score table in a game needs to record ten scores. One way to do this is to have a variable for each score:

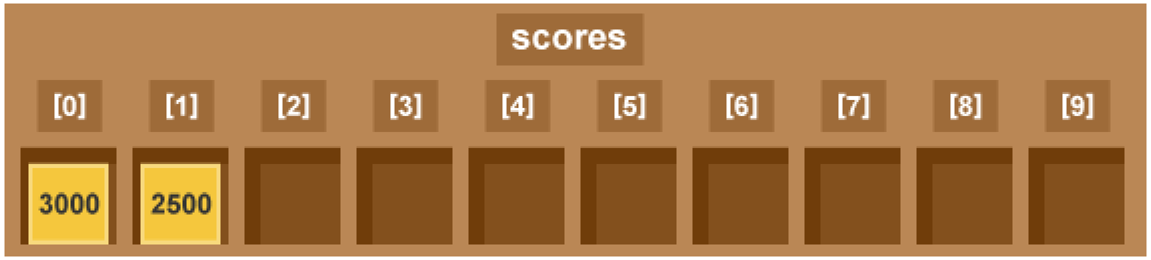
```
score_0
score_1
score_2
score_3
score_4
score_5
score_6
score_7
score_8
score_9
```

Instead of having ten variables, each holding a score, there could be one array that holds all the related data:

```
score(9)
```

All the data in an array must be of the **same data type**. In this example all the data items are integers.

The **size** of an array is declared when the program is **written**.

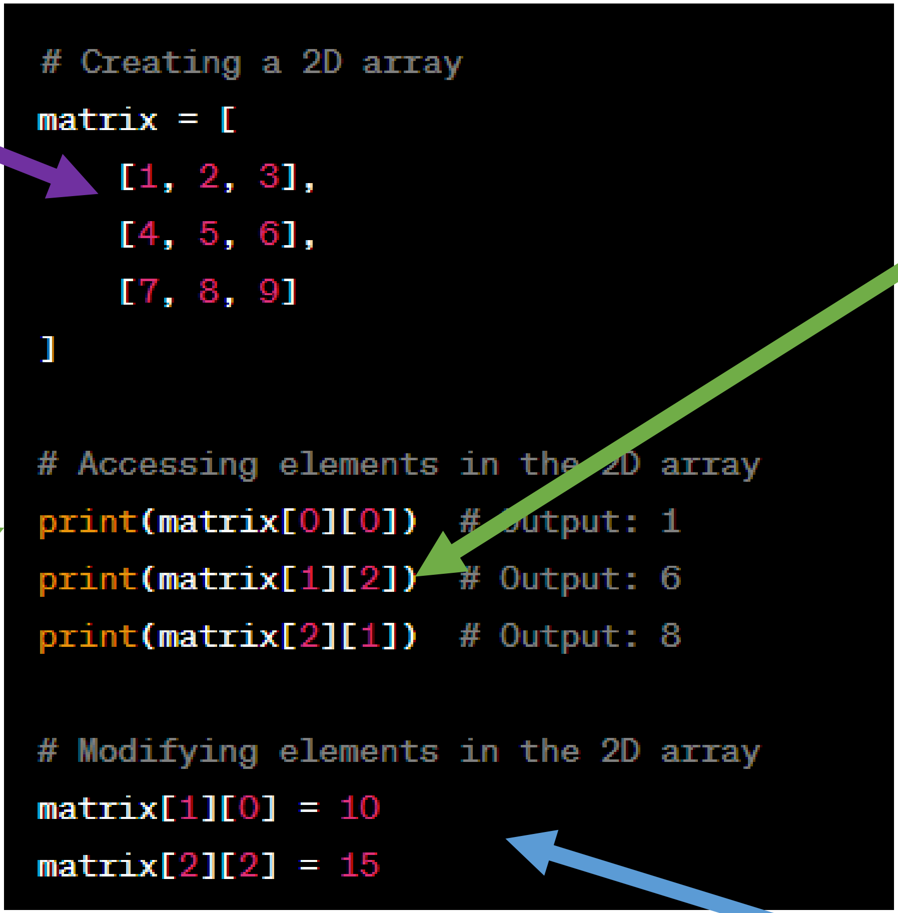


J277 - 2.2 Programming fundamentals

2D Arrays

A 2D array is also known as a matrix (a table of rows and columns).
To create a 2D array of integers, take a look at the following example:

In this example, we create a 2D array (a list of lists) called **matrix**. Each inner list represents a row, and the outer list represents the entire 2D array.



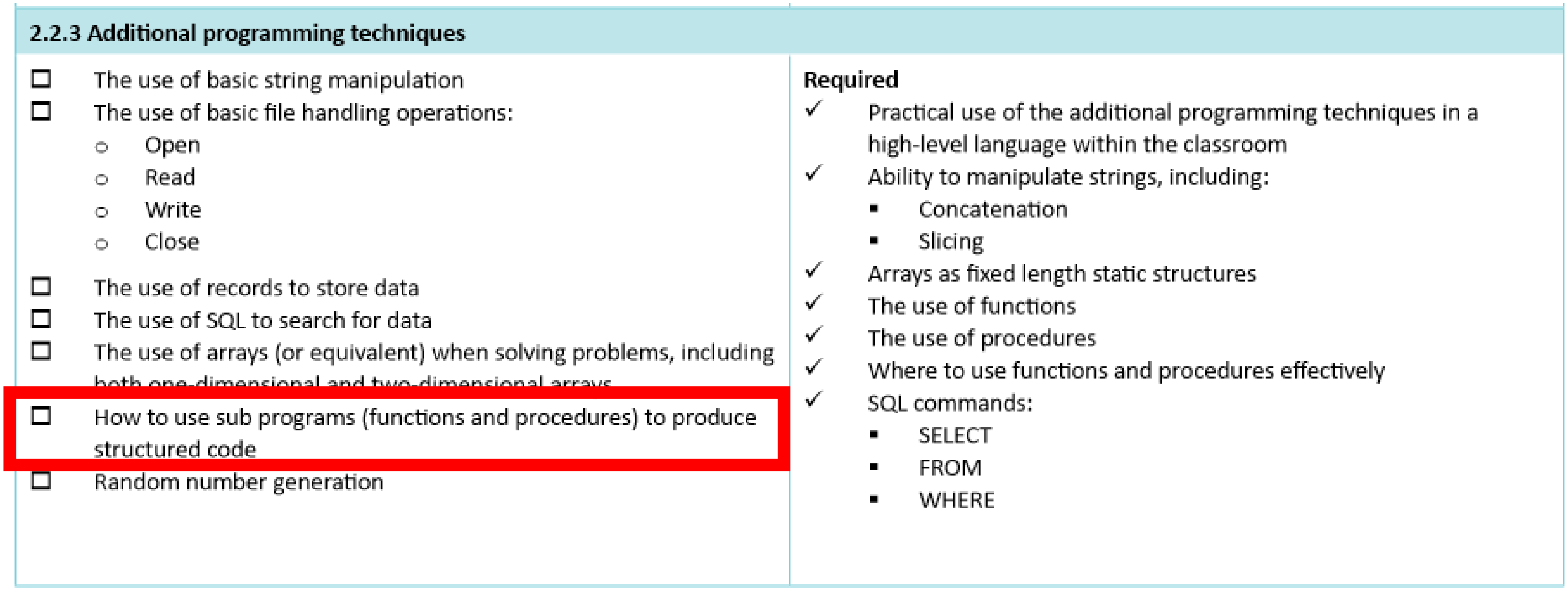
```
# Creating a 2D array
matrix = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]

# Accessing elements in the 2D array
print(matrix[0][0])  # Output: 1
print(matrix[1][2])  # Output: 6
print(matrix[2][1])  # Output: 8

# Modifying elements in the 2D array
matrix[1][0] = 10
matrix[2][2] = 15
```

Similarly, **matrix[1][2]** refers to the element at the second row and third column, which is 6.

We can access individual elements of the 2D array using indices. For example, **matrix[0][0]** refers to the element at the first row and first column, which is 1.

We can modify elements of the 2D array by assigning new values to specific indices. In the example, we change **matrix[1][0]** to 10 and **matrix[2][2]** to 15.

J277 - 2.2 Programming fundamentals



2.2.3 Additional programming techniques

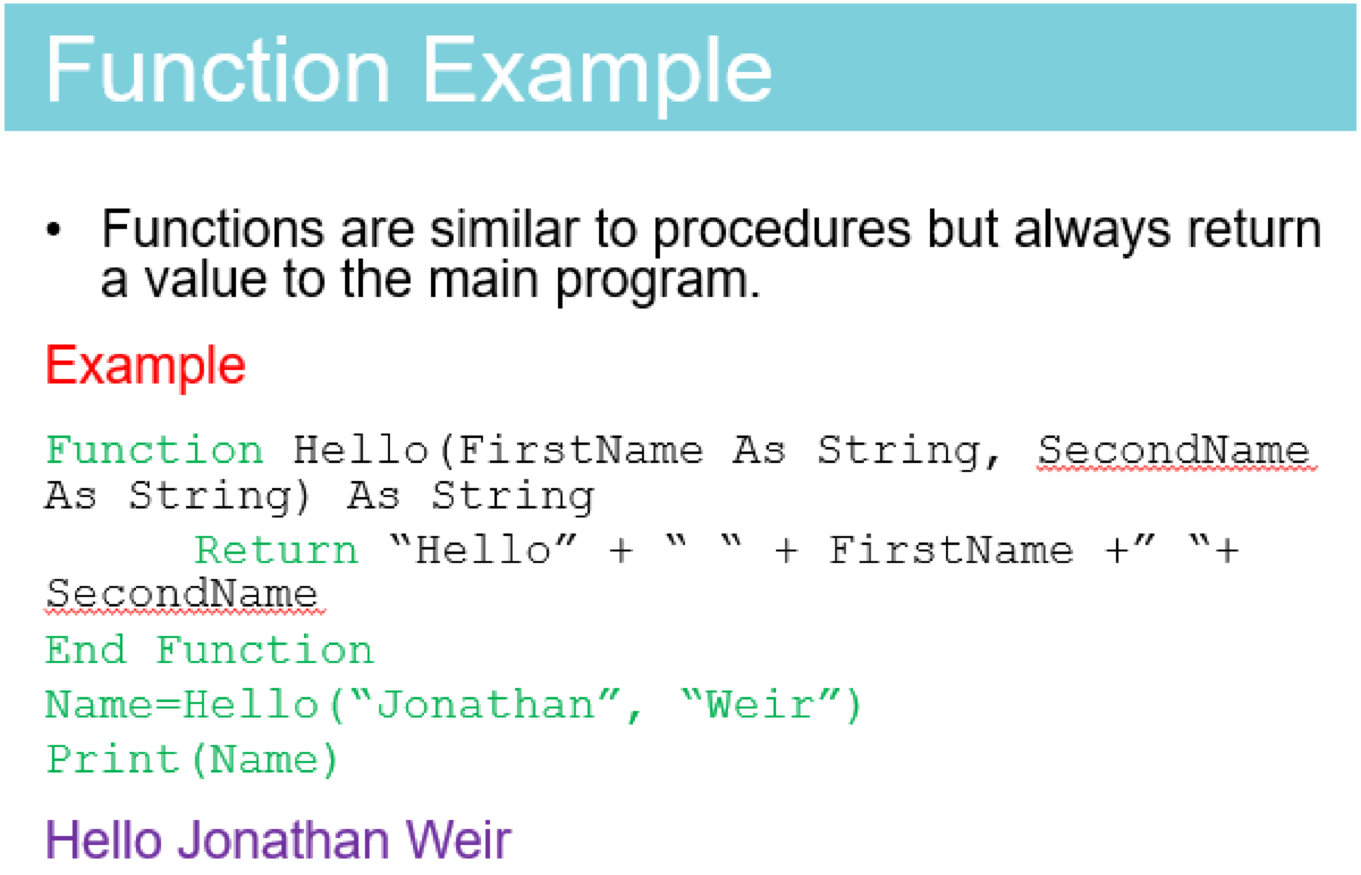
- ☐ The use of basic string manipulation
- ☐ The use of basic file handling operations:
  - ○ Open
  - ○ Read
  - ○ Write
  - ○ Close
- ☐ The use of records to store data
- ☐ The use of SQL to search for data
- ☐ The use of arrays (or equivalent) when solving problems, including both one-dimensional and two-dimensional arrays
- ☐ How to use sub programs (functions and procedures) to produce structured code
- ☐ Random number generation

Required

- ✓ Practical use of the additional programming techniques in a high-level language within the classroom
- ✓ Ability to manipulate strings, including:
  - ▪ Concatenation
  - ▪ Slicing
- ✓ Arrays as fixed length static structures
- ✓ The use of functions
- ✓ The use of procedures
- ✓ Where to use functions and procedures effectively
- ✓ SQL commands:
  - ▪ SELECT
  - ▪ FROM
  - ▪ WHERE

Sub programs (functions and procedures) to produce structured code

A function:	A sub program that **DOES** return a value.
The purpose of a function:	To create a reusable program component. Returning a value to be used in the program.



Function Example

- Functions are similar to procedures but always return a value to the main program.

Example

```
Function Hello(FirstName As String, SecondName
As String) As String
     Return "Hello" + " " + FirstName +" "+
SecondName
End Function
Name=Hello("Jonathan", "Weir")
Print(Name)
```

Hello Jonathan Weir

J277 - 2.2 Programming fundamentals

Sub programs (functions and procedures) to produce structured code

A procedure:	A sub program that **DOESN'T** return a value.
The purpose of a procedure:	To produce structured code that is easier to read and debug.

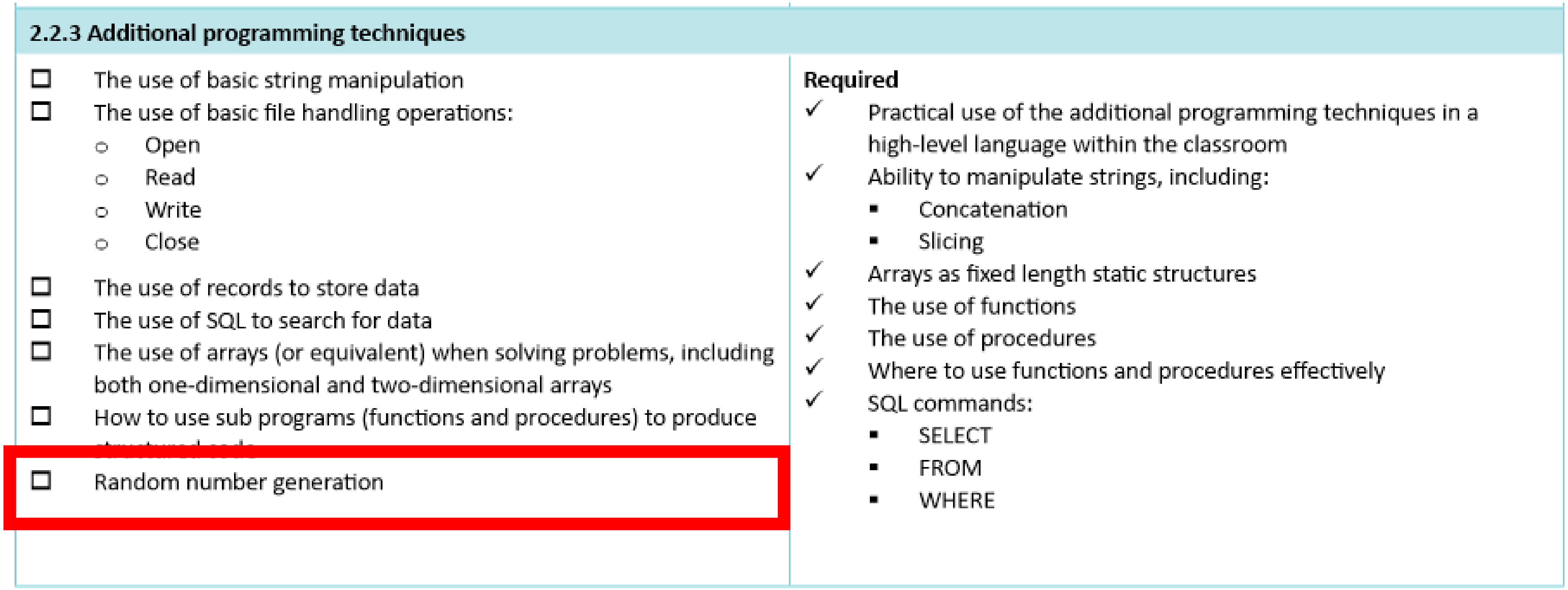
Procedure example

- Procedures are sets of instructions stored under one name (identifier).

```
Example
Procedure name()
      name=Input("What is your name?")
      print("Hello " + name)
End Procedure
```

e.g. if we call name() and input Jon it would output:
Hello Jon

J277 - 2.2 Programming fundamentals



2.2.3 Additional programming techniques

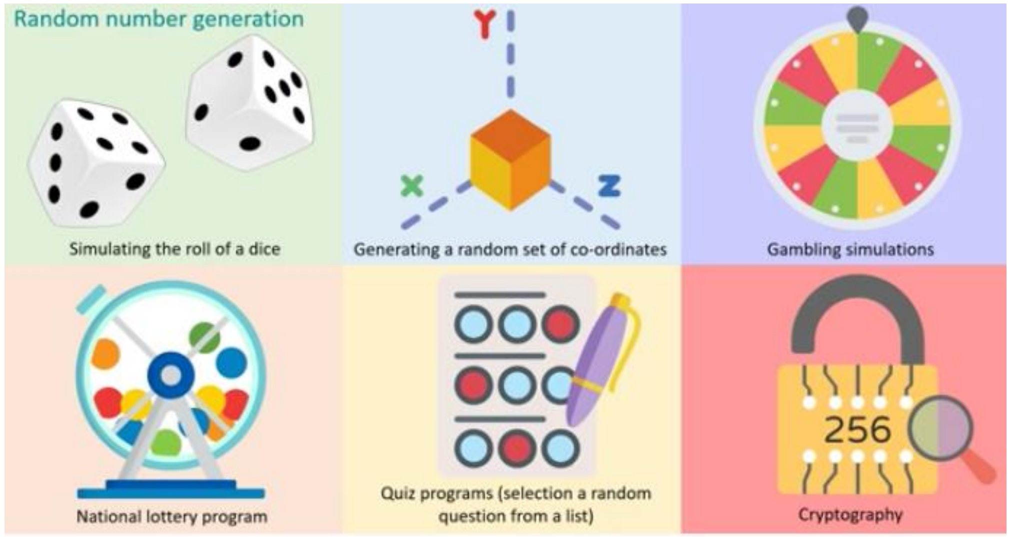
- ☐ The use of basic string manipulation
- ☐ The use of basic file handling operations:
  - Open
  - Read
  - Write
  - Close
- ☐ The use of records to store data
- ☐ The use of SQL to search for data
- ☐ The use of arrays (or equivalent) when solving problems, including both one-dimensional and two-dimensional arrays
- ☐ How to use sub programs (functions and procedures) to produce structured code
- ☐ Random number generation

Required

- ✓ Practical use of the additional programming techniques in a high-level language within the classroom
- ✓ Ability to manipulate strings, including:
  - Concatenation
  - Slicing
- ✓ Arrays as fixed length static structures
- ✓ The use of functions
- ✓ The use of procedures
- ✓ Where to use functions and procedures effectively
- ✓ SQL commands:
  - SELECT
  - FROM
  - WHERE

J277 - 2.2 Programming fundamentals

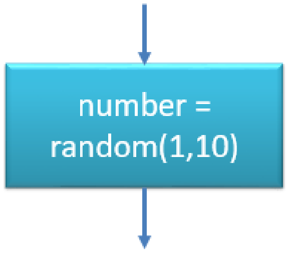
Random number generation


Random number generation
Simulating the roll of a dice
Generating a random set of co-ordinates
Y
X
Z
Gambling simulations
National lottery program
Quiz programs (selection a random question from a list)
256
Cryptography

Generating random numbers

- A typical method of generating random numbers might look like this:

```
month:= getRandBetween(1,12)
```



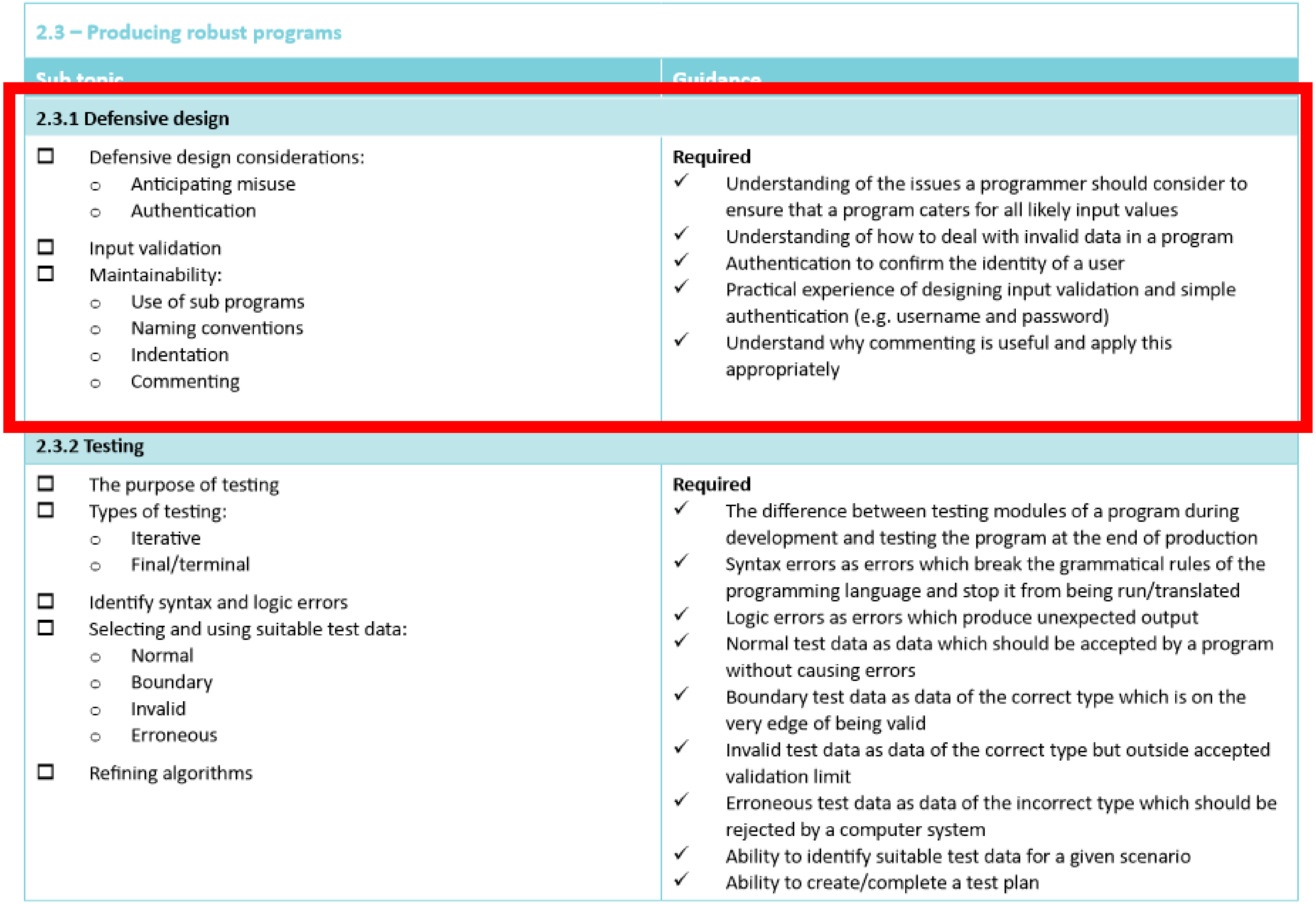
```
import random
diceScore = random.randint(1,6)
```

Syntax for exam!

- The exact structure of a random number generator will vary between programming languages.
- Using the OCR Exam Reference Language you can either state the first and last possible values
  - *e.g. diceRoll = random(1,6)*
- Or just the maximum value, if starting from 1
  - *e.g. diceRoll = random(6)*

```
import random
diceScore = random.randint(1,6)
```

J277 - 2.3 Producing robust programs



2.3 – Producing robust programs

Sub topic	Guidance
2.3.1 Defensive design	
☐ Defensive design considerations: ○ Anticipating misuse ○ Authentication ☐ Input validation ☐ Maintainability: ○ Use of sub programs ○ Naming conventions ○ Indentation ○ Commenting	**Required** ✓ Understanding of the issues a programmer should consider to ensure that a program caters for all likely input values ✓ Understanding of how to deal with invalid data in a program ✓ Authentication to confirm the identity of a user ✓ Practical experience of designing input validation and simple authentication (e.g. username and password) ✓ Understand why commenting is useful and apply this appropriately
2.3.2 Testing	
☐ The purpose of testing ☐ Types of testing: ○ Iterative ○ Final/terminal ☐ Identify syntax and logic errors ☐ Selecting and using suitable test data: ○ Normal ○ Boundary ○ Invalid ○ Erroneous ☐ Refining algorithms	**Required** ✓ The difference between testing modules of a program during development and testing the program at the end of production ✓ Syntax errors as errors which break the grammatical rules of the programming language and stop it from being run/translated ✓ Logic errors as errors which produce unexpected output ✓ Normal test data as data which should be accepted by a program without causing errors ✓ Boundary test data as data of the correct type which is on the very edge of being valid ✓ Invalid test data as data of the correct type but outside accepted validation limit ✓ Erroneous test data as data of the incorrect type which should be rejected by a computer system ✓ Ability to identify suitable test data for a given scenario ✓ Ability to create/complete a test plan

J277 - 2.3 Producing robust programs

Robust programming

Programs that function correctly shouldn't break or produce errors. Avoid these problems by using defensive design:

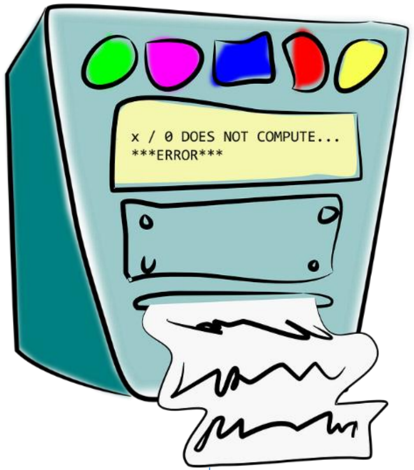
Anticipate and prevent misuse by users.

Keep code well-maintained.

Reduce errors by testing.

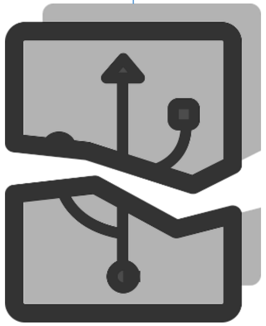
Defensive design: Anticipating misuse

Even with valid inputs there are a number of reasons why a program could crash. These should be trapped by the programmer with exception handling code.



Division by zero.

Communication error.
E.g. lost connection to host.



Printer out of paper.
Printer out of ink.
Paper jam.

Out of disk space.
File not found.
End of file.
Invalid data in file.

A user might also misinterpret the on-screen prompts, or enter data into the wrong input box. A programmer should plan for all possible eventualities.

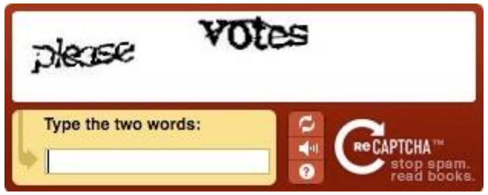
J277 - 2.3 Producing robust programs

Defensive design: Authentication

Confirming the identity of a user before allowing access. Passwords or biometrics are usually associated with a username.

The most common way of doing this is to ask for a **username** and **password**. The entered details are then checked against a database containing valid accounts.

reCaptcha is a method used to protect online forms against bots. A bot can automatically submit data in online forms creating spam. A robust program needs to identify the user as a human and not another program. Often the user has to type the words they see on-screen which are presented as pictures in a format that would be difficult for a program to decipher, instead of text.



For high security sites such as financial services **'two-factor' authentication** is becoming popular. This means that after the user enters a valid user name and password, the system sends an SMS text 'authentication code' to the designated mobile phone. They then have to enter this as well.

Biometrics is method that checks some physical feature of the authorised person such as their fingerprint. The user puts their thumb on a fingerprint entry device, the data is sent off to a database and checked against their valid data.

J277 - 2.3 Producing robust programs

Defensive design: Input validation

Input validation:	Checking if data meets certain criteria before passing it into a program.

Check digit	the last one or two digits in a code are used to check the other digits are correct	bar code readers in supermarkets use check digits
Format check	checks the data is in the right format	a National Insurance number is in the form LL 99 99 99 L where L is any letter and 9 is any number
Length check	checks the data isn't too short or too long	a password which needs to be six letters long
Lookup table	looks up acceptable values in a table	there are only seven possible days of the week
Presence check	checks that data has been entered into a field	in most databases a key field cannot be left blank
Range check	checks that a value falls within the specified range	number of hours worked must be less than 50 and more than 0
Spell check	looks up words in a dictionary	MS Word uses red lines to underline misspelt words

J277 - 2.3 Producing robust programs

Maintainability

Well maintained code is easier for other programmers to understand. They can change parts of the code without causing problems elsewhere.

1. #Write **comments** to explain what is happening at each stage. #ensures the input is restricted to a y or an n

```
do
    for count = 1 to 10
         print("Hey!")
    next count
    again = input("Enter y to go again")
    again.lower
until again !="y"
```

2. Use **indentation** to make the program flow easier to see and show selection and iteration code branches.

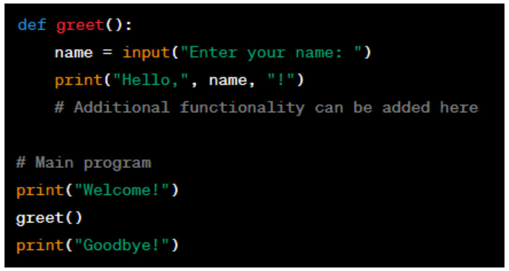
3. Use of **whitespace** to easily see where functions begin and end.

H - poor choice. What does it mean?

height - better choice. The value to be inputted will be a number.

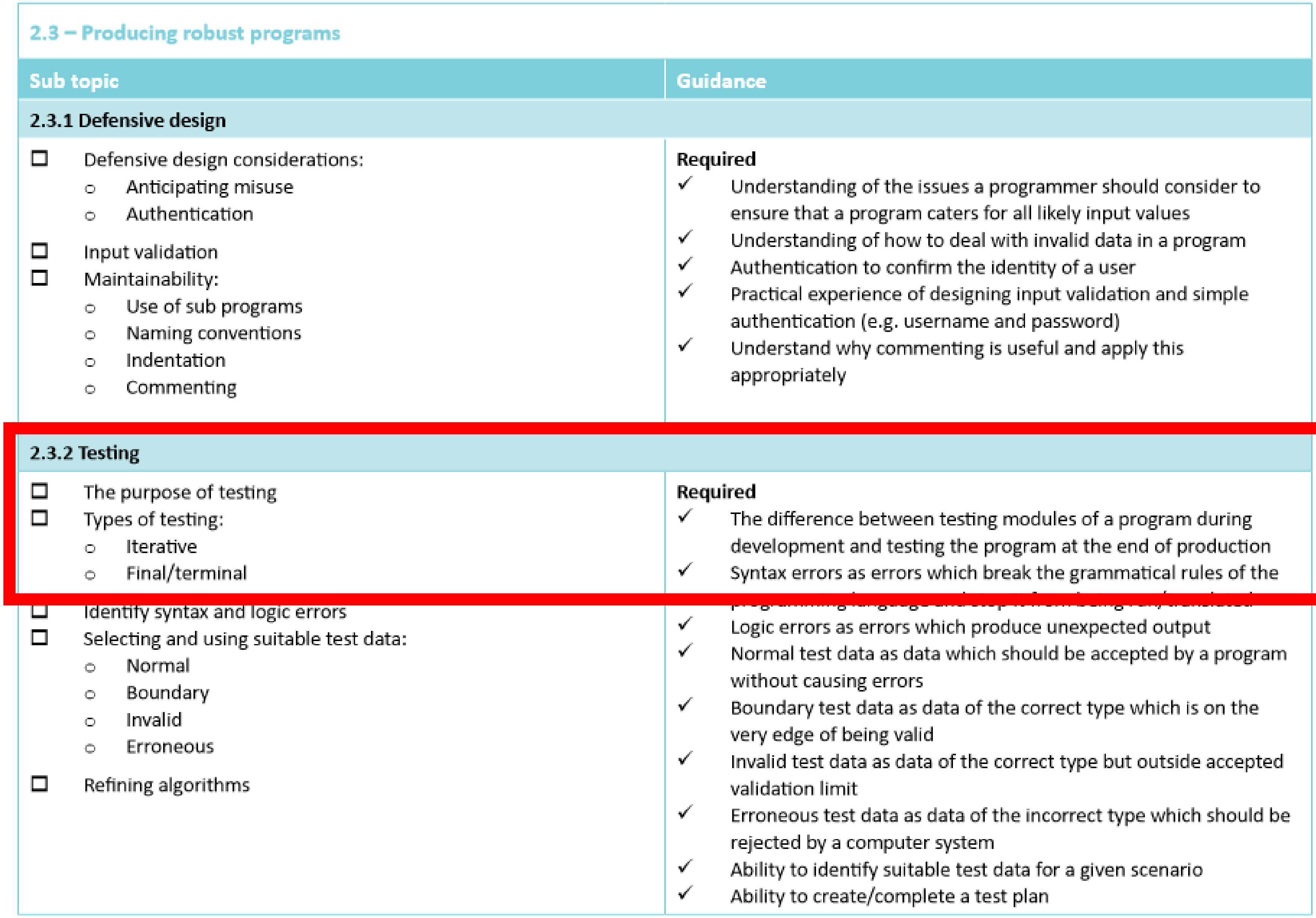
4. Descriptive **variable** and **function** names.

5. Use of **sub-programs** to separate parts of the program.



```
def greet():
    name = input("Enter your name: ")
    print("Hello,", name, "!")
    # Additional functionality can be added here

# Main program
print("Welcome!")
greet()
print("Goodbye!")
```

J277 - 2.3 Producing robust programs



2.3 – Producing robust programs

Sub topic	Guidance
2.3.1 Defensive design	
☐ Defensive design considerations: ○ Anticipating misuse ○ Authentication ☐ Input validation ☐ Maintainability: ○ Use of sub programs ○ Naming conventions ○ Indentation ○ Commenting	**Required** ✓ Understanding of the issues a programmer should consider to ensure that a program caters for all likely input values ✓ Understanding of how to deal with invalid data in a program ✓ Authentication to confirm the identity of a user ✓ Practical experience of designing input validation and simple authentication (e.g. username and password) ✓ Understand why commenting is useful and apply this appropriately
2.3.2 Testing	
☐ The purpose of testing ☐ Types of testing: ○ Iterative ○ Final/terminal ☐ Identify syntax and logic errors ☐ Selecting and using suitable test data: ○ Normal ○ Boundary ○ Invalid ○ Erroneous ☐ Refining algorithms	**Required** ✓ The difference between testing modules of a program during development and testing the program at the end of production ✓ Syntax errors as errors which break the grammatical rules of the programming language and stop it from being run/translated ✓ Logic errors as errors which produce unexpected output ✓ Normal test data as data which should be accepted by a program without causing errors ✓ Boundary test data as data of the correct type which is on the very edge of being valid ✓ Invalid test data as data of the correct type but outside accepted validation limit ✓ Erroneous test data as data of the incorrect type which should be rejected by a computer system ✓ Ability to identify suitable test data for a given scenario ✓ Ability to create/complete a test plan

J277 - 2.3 Producing robust programs

The purpose and types of testing

Four main reasons why a program should be thoroughly tested before being given to a user:

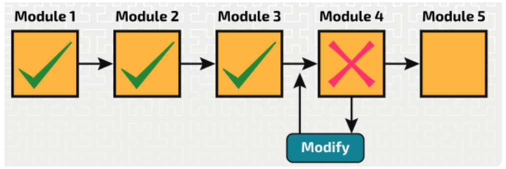
To check the program meets the requirements.

To ensure there are no logic errors.

To check the program has an acceptable performance and usability.

Ensure unauthorised access is prevented.

Iterative Testing

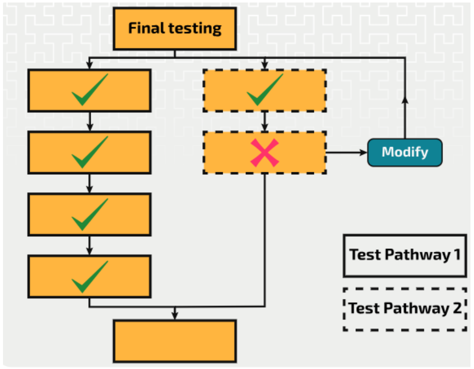


Performed whilst the software is being developed.

The programmer writes an individual section of code or part of the project and then tests it to ensure that it functions correctly. It is likely that errors will be detected, and these are resolved before the next section of the program is developed and tested.

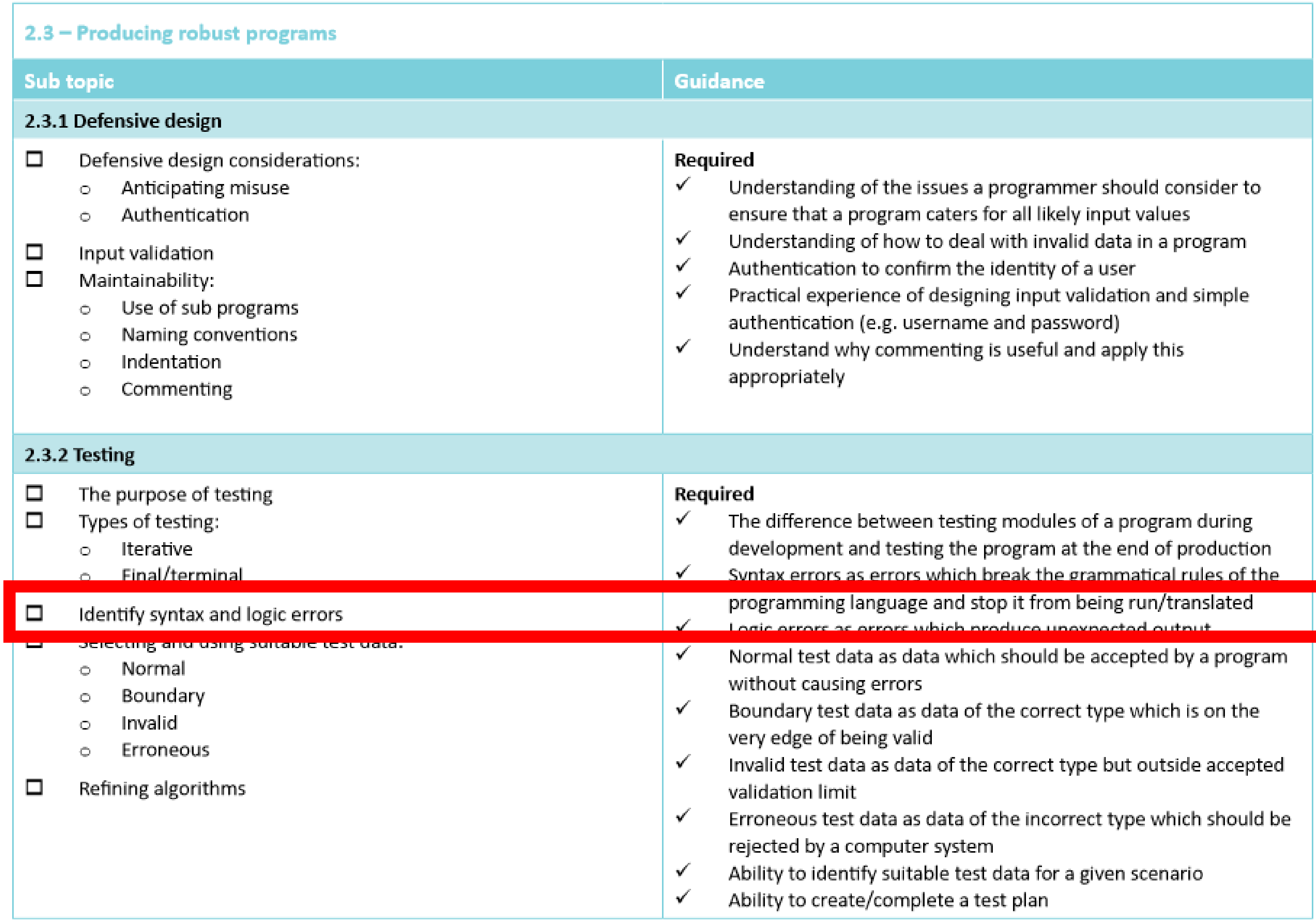
Final/Terminal Testing

Performed at the end of development



This stage happens after the individual sections or modules of the system have been tested, to ensure that the system works as a whole and that it meets the requirements of the project.

J277 - 2.3 Producing robust programs



2.3 – Producing robust programs

Sub topic	Guidance
2.3.1 Defensive design	
☐ Defensive design considerations: ○ Anticipating misuse ○ Authentication ☐ Input validation ☐ Maintainability: ○ Use of sub programs ○ Naming conventions ○ Indentation ○ Commenting	**Required** ✓ Understanding of the issues a programmer should consider to ensure that a program caters for all likely input values ✓ Understanding of how to deal with invalid data in a program ✓ Authentication to confirm the identity of a user ✓ Practical experience of designing input validation and simple authentication (e.g. username and password) ✓ Understand why commenting is useful and apply this appropriately
2.3.2 Testing	
☐ The purpose of testing ☐ Types of testing: ○ Iterative ○ Final/terminal ☐ Identify syntax and logic errors ☐ Selecting and using suitable test data: ○ Normal ○ Boundary ○ Invalid ○ Erroneous ☐ Refining algorithms	**Required** ✓ The difference between testing modules of a program during development and testing the program at the end of production ✓ Syntax errors as errors which break the grammatical rules of the programming language and stop it from being run/translated ✓ Logic errors as errors which produce unexpected output ✓ Normal test data as data which should be accepted by a program without causing errors ✓ Boundary test data as data of the correct type which is on the very edge of being valid ✓ Invalid test data as data of the correct type but outside accepted validation limit ✓ Erroneous test data as data of the incorrect type which should be rejected by a computer system ✓ Ability to identify suitable test data for a given scenario ✓ Ability to create/complete a test plan

J277 - 2.3 Producing robust programs

How to identify syntax errors

A syntax error occurs when code written does not follow the rules of the programming language. Examples include:

- Misspelling a statement, eg writing `pint` instead of `print`
- Using a **variable** before it has been declared
- Missing **brackets**, eg opening a bracket but not closing it

```
# Example: Syntax Error
print("Hello, World!"
```

```
# Correction
print("Hello, World!")
```

How to identify logic errors

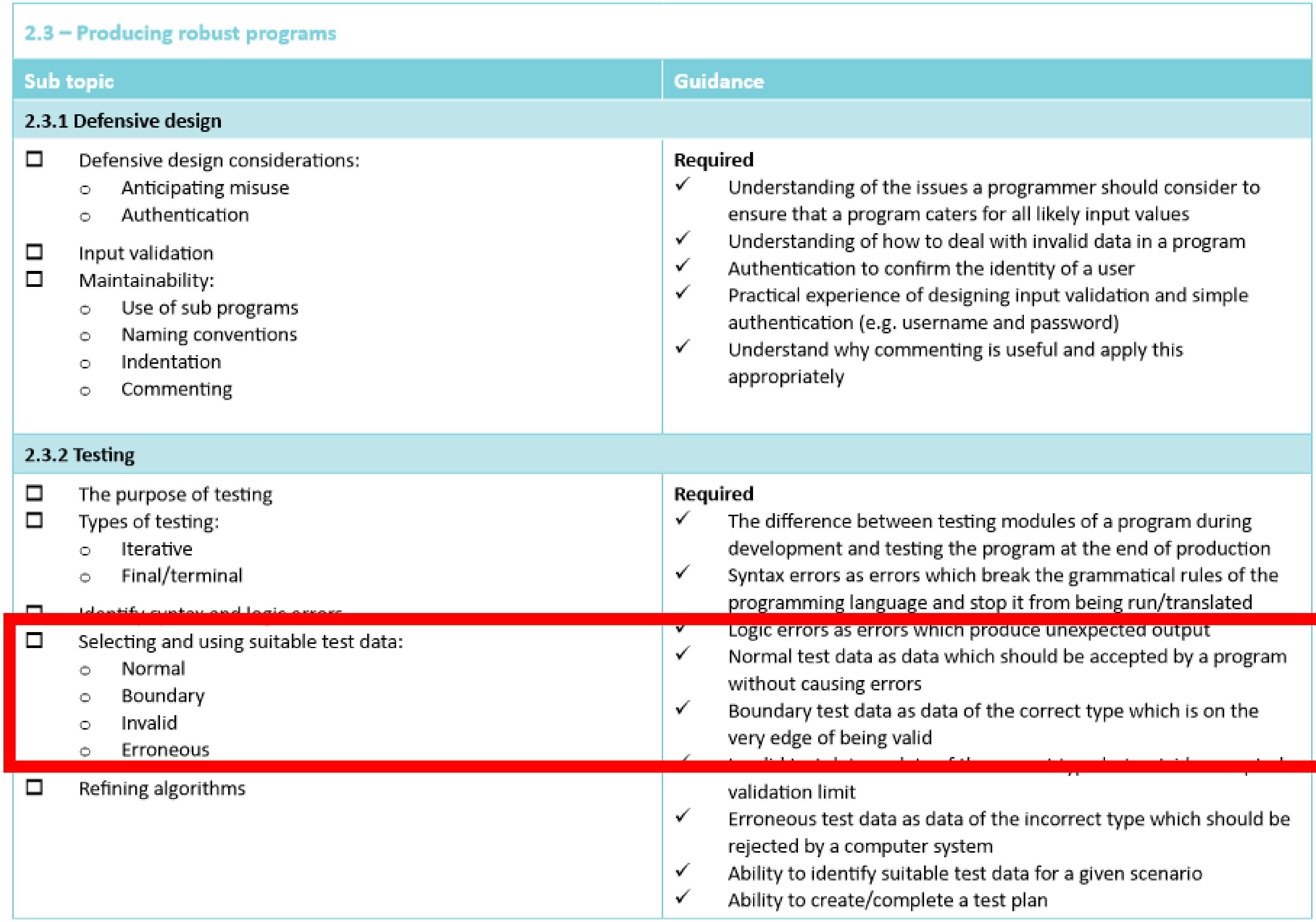
A logic error is an error in the way a program works. The program simply does not do what it is expected to do.

- Logic errors can have many causes, such as:
- Incorrectly using logical operators, eg expecting a program to stop when the value of a variable reaches 5, but using <5 instead of <=5
- Incorrectly using **Boolean operators**
- Unintentionally creating a situation where an **infinite loop** may occur
- Incorrectly using brackets in calculations
- Unintentionally using the same variable name at different points in the program for different purposes
- Referring to an element in an **array** that falls outside of the scope of the array

```
# Example: Logic Error
num1 = 5
num2 = 3
sum = num1 - num2  # Incorrect operation

print("The sum of", num1, "and", num2, "is:", sum)
```

```
# Correction
sum = num1 + num2
```

J277 - 2.3 Producing robust programs



2.3 – Producing robust programs

Sub topic	Guidance
2.3.1 Defensive design	
☐ Defensive design considerations: ○ Anticipating misuse ○ Authentication ☐ Input validation ☐ Maintainability: ○ Use of sub programs ○ Naming conventions ○ Indentation ○ Commenting	**Required** ✓ Understanding of the issues a programmer should consider to ensure that a program caters for all likely input values ✓ Understanding of how to deal with invalid data in a program ✓ Authentication to confirm the identity of a user ✓ Practical experience of designing input validation and simple authentication (e.g. username and password) ✓ Understand why commenting is useful and apply this appropriately
2.3.2 Testing	
☐ The purpose of testing ☐ Types of testing: ○ Iterative ○ Final/terminal ☐ Identify syntax and logic errors ☐ Selecting and using suitable test data: ○ Normal ○ Boundary ○ Invalid ○ Erroneous ☐ Refining algorithms	**Required** ✓ The difference between testing modules of a program during development and testing the program at the end of production ✓ Syntax errors as errors which break the grammatical rules of the programming language and stop it from being run/translated ✓ Logic errors as errors which produce unexpected output ✓ Normal test data as data which should be accepted by a program without causing errors ✓ Boundary test data as data of the correct type which is on the very edge of being valid ✓ Invalid test data as data of the correct type but outside accepted validation limit ✓ Erroneous test data as data of the incorrect type which should be rejected by a computer system ✓ Ability to identify suitable test data for a given scenario ✓ Ability to create/complete a test plan

J277 - 2.3 Producing robust programs

Selecting and using suitable test data

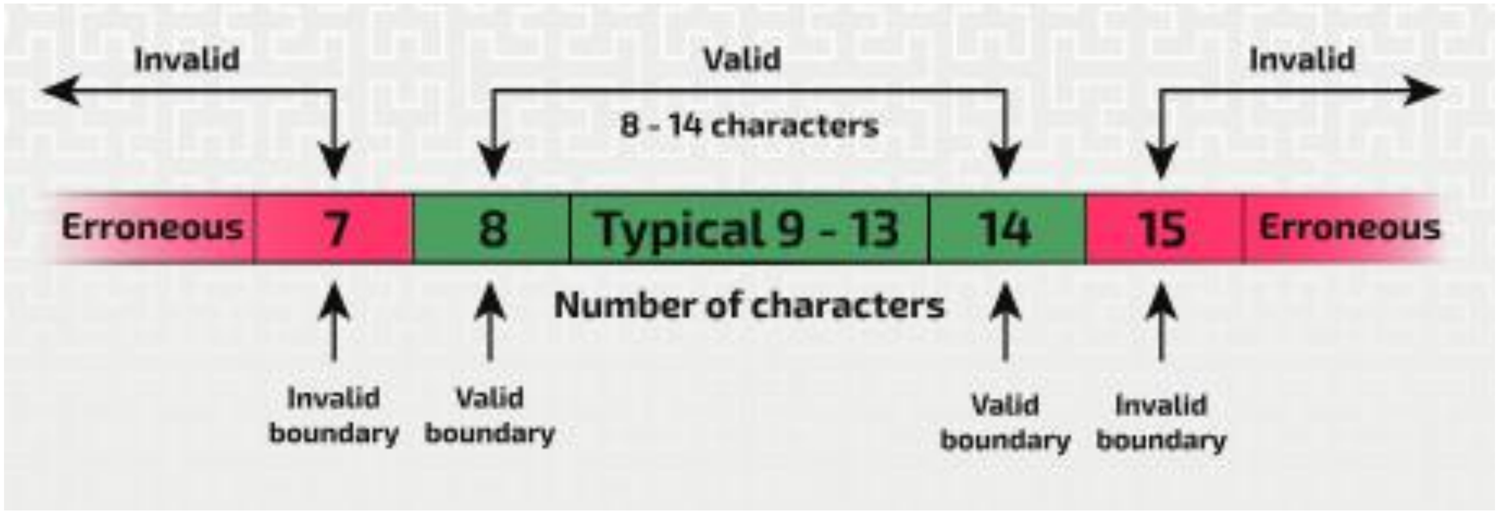
Test data is data that is used to test whether or not a program is functioning correctly. Ideally, test data should cover a range of possible and impossible inputs, each designed to prove a program works or to highlight any flaws. Three types of test data are:

Normal data - typical, sensible data that the program should accept and be able to process.

•**Boundary data** - valid data that falls at the boundary of any possible ranges, sometimes known as extreme data.

•**Erroneous data** - data that the program cannot process and should not accept.

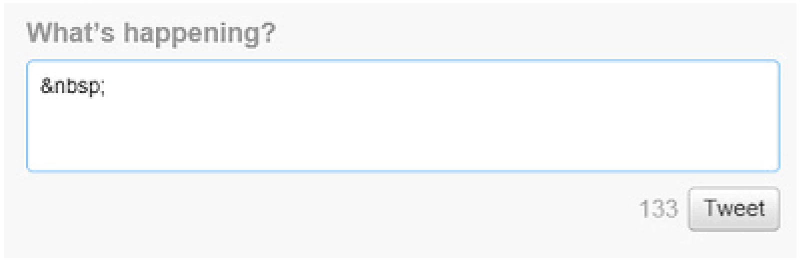
Types of test data for a password checking system:



J277 - 2.3 Producing robust programs

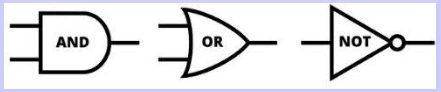
Selecting and using suitable test data

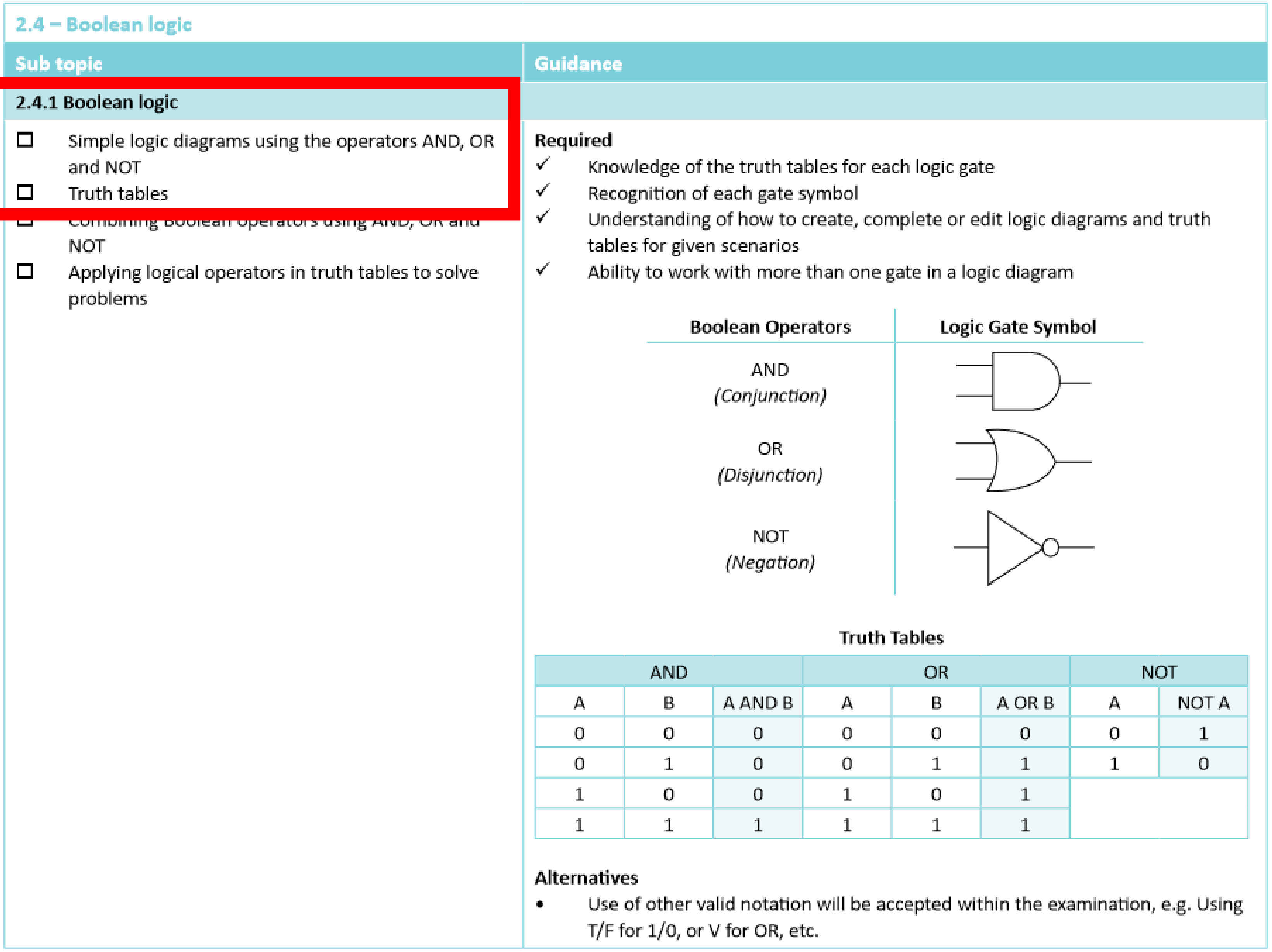
In September 2017, Twitter announced it was testing doubling the number of characters in a tweet from 140 to 280 characters. Twitter's character limit is a holdover from the app's early days when tweets were sent as texts, which were limited to 160 characters. It has since become one of the product's defining characteristics. A typical test table that could be used:



Test No.	No. characters input	Type of test	Reason for the test
1	67	Normal data	Check the new functionality doesn't break the original working code. Check the character counter updates correctly.
2	280	Boundary data	Check the maximum number of characters. Check the character counter updates correctly to 0.
3	281	Invalid data	Check only tweets up to 280 characters are accepted. Check the character counter updates correctly to negative number.
4	0	Invalid data	A blank tweet should not be stored.
5	1	Boundary data	Check the minimum number of characters.

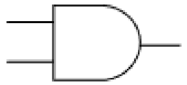
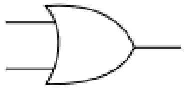
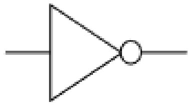
J277 - 2.4 Boolean logic





2.4 – Boolean logic

Sub topic	Guidance
2.4.1 Boolean logic	
☐ Simple logic diagrams using the operators AND, OR and NOT ☐ Truth tables ☐ Combining Boolean operators using AND, OR and NOT ☐ Applying logical operators in truth tables to solve problems	**Required** ✓ Knowledge of the truth tables for each logic gate ✓ Recognition of each gate symbol ✓ Understanding of how to create, complete or edit logic diagrams and truth tables for given scenarios ✓ Ability to work with more than one gate in a logic diagram

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

Truth Tables

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

Alternatives

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

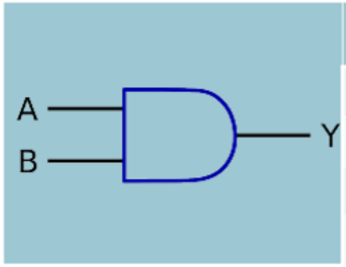
J277 - 2.4 Boolean logic

Simple logic diagrams using the operations AND, OR and NOT

Used to change bits and perform calculations within a computer. They are created by using transistors.
There are 3 basic logic gates:

The AND Gate

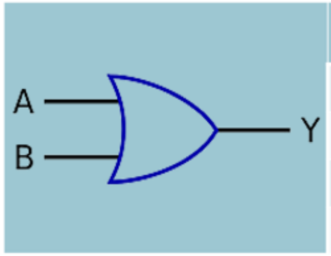
- Only has an output of 1 if both A AND B are 1
- We write this as: **A ∧ B**



A	B	A ∧ B or Y
0	0	0
0	1	0
1	0	0
1	1	1

The OR Gate

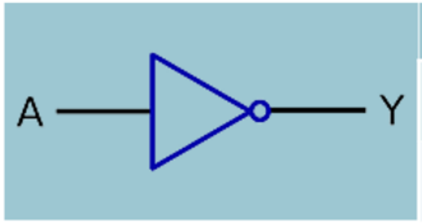
- Has an output of 1 if either of A OR B are 1
- We write this as: **A** V **B**



A	B	A V B or Y
0	0	0
0	1	1
1	0	1
1	1	1

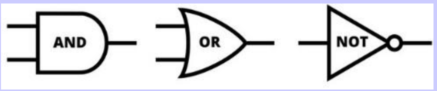
The NOT Gate

- Has an 1 input only
- It **INVERTS** or swaps the input
- We write this as: ¬ **A**



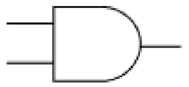
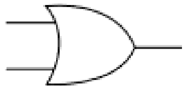
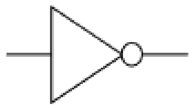
A	¬ A or Y
0	1
1	0

J277 - 2.4 Boolean logic



2.4 – Boolean logic

Sub topic	Guidance
2.4.1 Boolean logic	
☐ Simple logic diagrams using the operators AND, OR and NOT ☐ Truth tables ☐ Combining Boolean operators using AND, OR and NOT ☐ Applying logical operators in truth tables to solve problems	**Required** ✓ Knowledge of the truth tables for each logic gate ✓ Recognition of each gate symbol ✓ Understanding of how to create, complete or edit logic diagrams and truth tables for given scenarios ✓ Ability to work with more than one gate in a logic diagram

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

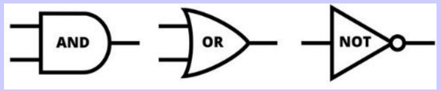
Truth Tables

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

Alternatives

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

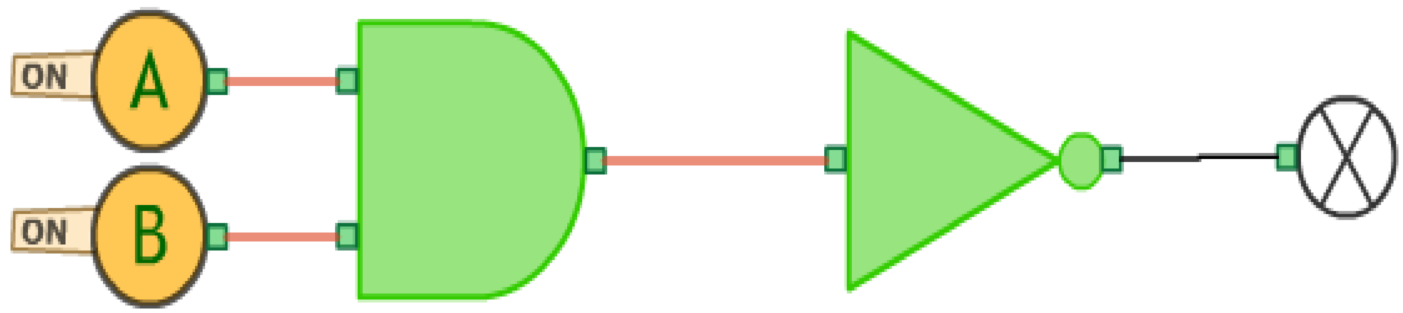
J277 - 2.4 Boolean logic



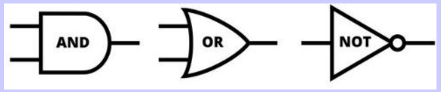
Combining Boolean operators using AND, OR and NOT to two levels

Logic diagram for the expression: NOT (A AND B)

Alternative notation : ¬ **(A ∧ B)**



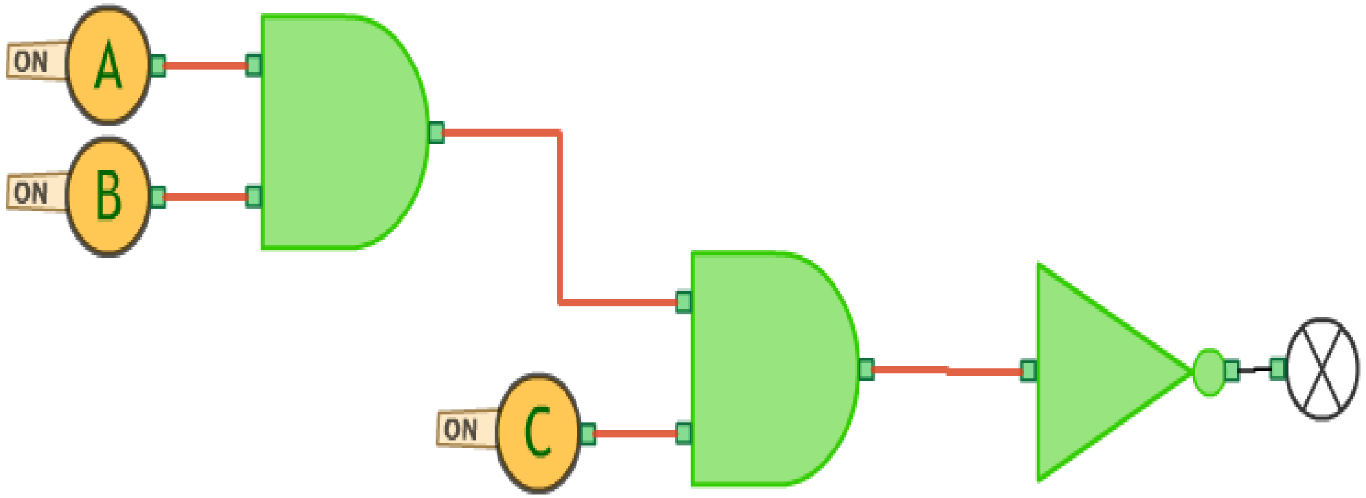
J277 - 2.4 Boolean logic



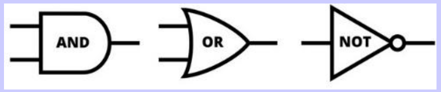
Combining Boolean operators using AND, OR and NOT to two levels

Logic diagram for the expression: NOT(A AND B) AND C)

Alternative notation : ¬ **((A ⋀ B) ⋀ C)**



J277 - 2.4 Boolean logic



Applying logical operators in truth tables to solve problems

Draw the truth table for the following logic statement: **¬ (A ⋀ B)**

A	B	A ⋀ B	¬ (A ⋀ B)
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

Applying logical operators in truth tables to solve problems

Draw the truth table for the following logic statement: **¬ ((A ⋀ B) ⋀ C)**

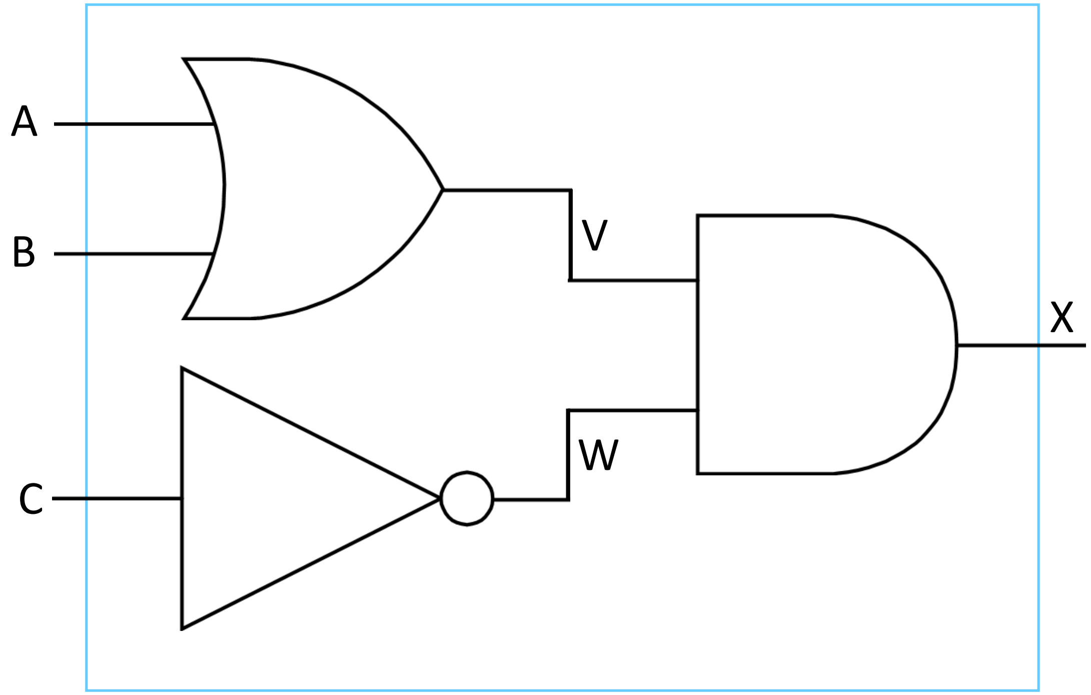
A	B	C	A ⋀ B	(A ⋀ B) ⋀ C)	¬ ((A ⋀ B) ⋀ C)
0	0	0	0	0	1
0	0	1	0	0	1
0	1	0	0	0	1
0	1	1	0	0	1
1	0	0	0	0	1
1	0	1	0	0	1
1	1	0	1	0	1
1	1	1	1	1	0

J277 - 2.4 Boolean logic

Create, complete or edit logic diagrams and truth tables for given scenarios

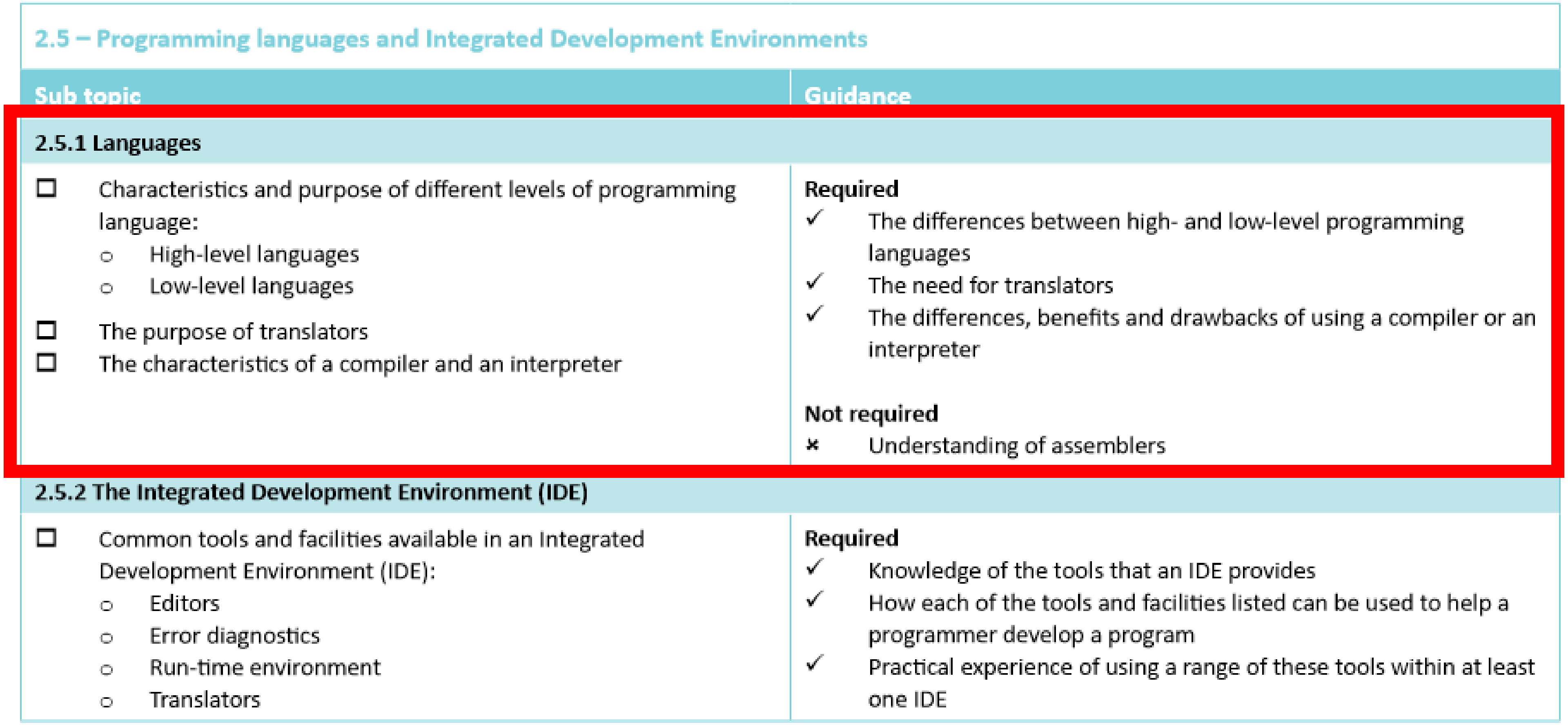
Logic diagram and truth table for the following scenario:

“A factory has an automated manufacturing system which operates in an “ON” state (OUTPUT X) if either it is manually switched on by an operator (INPUT A), OR a computer system triggers a scheduled production run (INPUT B). The system also has an emergency override (INPUT C) which in its normal operating state is feeding no signal to the computer system, when it is pressed however it triggers a positive “TRUE” state (1) which should result in the system shutting down.”



A	B	C	V = A OR B	W = NOT C	X = V AND W
0	0	0	0	1	0
0	0	1	0	0	0
0	1	0	1	1	1
0	1	1	1	0	0
1	0	0	1	1	1
1	0	1	1	0	0
1	1	0	1	1	1
1	1	1	1	0	0

J277 - 2.5 Programming languages and IDEs



2.5 – Programming languages and Integrated Development Environments

Sub topic	Guidance
2.5.1 Languages	
☐ Characteristics and purpose of different levels of programming language: ○ High-level languages ○ Low-level languages ☐ The purpose of translators ☐ The characteristics of a compiler and an interpreter	**Required** ✓ The differences between high- and low-level programming languages ✓ The need for translators ✓ The differences, benefits and drawbacks of using a compiler or an interpreter **Not required** ✗ Understanding of assemblers
2.5.2 The Integrated Development Environment (IDE)	
☐ Common tools and facilities available in an Integrated Development Environment (IDE): ○ Editors ○ Error diagnostics ○ Run-time environment ○ Translators	**Required** ✓ Knowledge of the tools that an IDE provides ✓ How each of the tools and facilities listed can be used to help a programmer develop a program ✓ Practical experience of using a range of these tools within at least one IDE

J277 - 2.5 Programming languages and IDEs

Characteristics and purpose of different levels of programming language

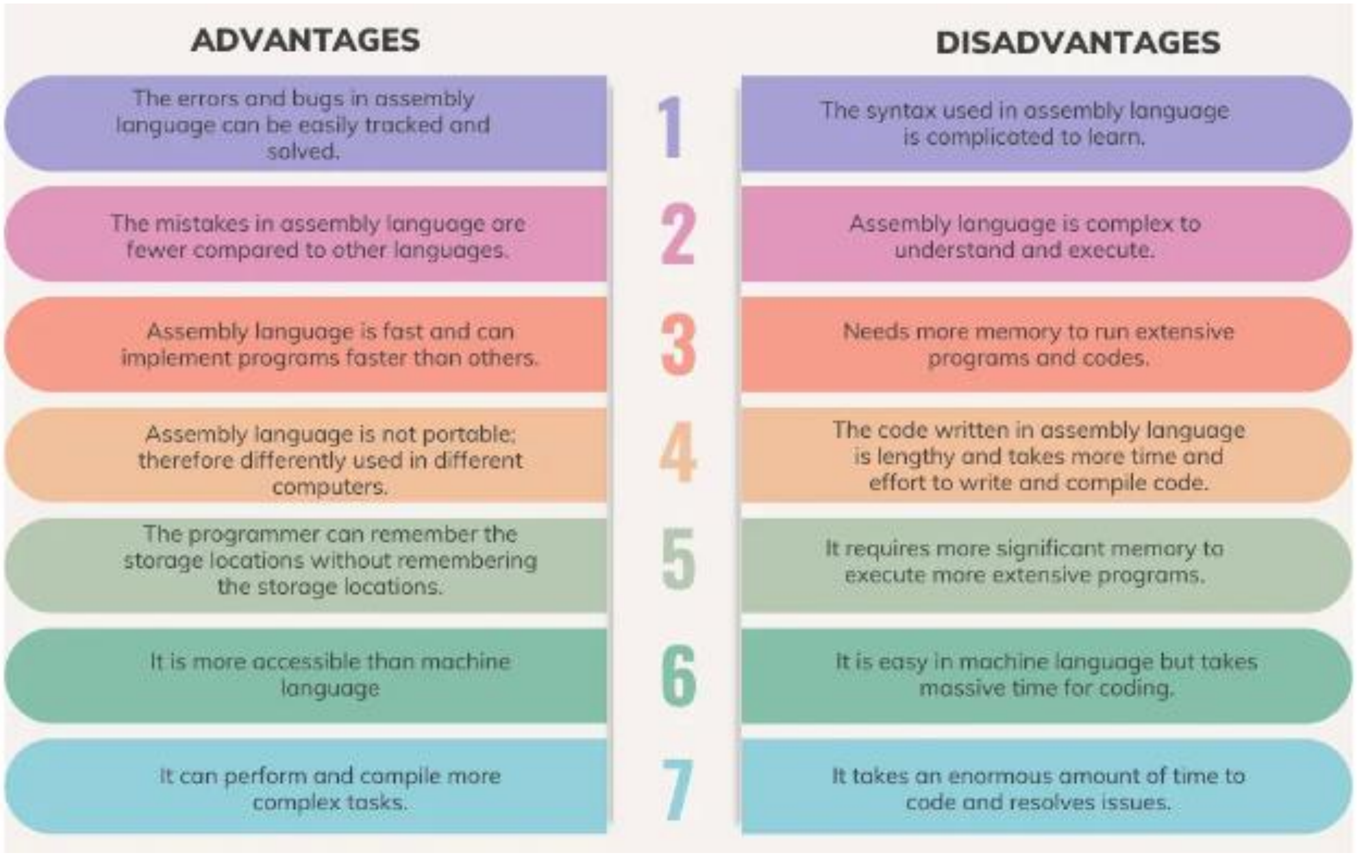
	High level language	**Low level language**
Example language:	Python	Assembly
Purpose:	Makes writing of computer programs easier by using commands that are similar to English language.	Used for embedded systems and device drivers where instructing the hardware directly is necessary.
Characteristic 1 Instructions to machine code:	One instruction translates to many machine code instructions.	One instruction translates to one machine code instruction.
Characteristic 2 Types of processors:	Code will run on different processors.	The code will work on one type of processor.
Characteristic 3 Data structures:	The programmer has lots of data structures to use.	The programmer works with the memory directly.
Characteristic 4 Ease of coding:	Code is quicker and easier to understand and write.	Code is much harder to understand and write.
Characteristic 5 Memory efficiency:	Less memory efficient.	More memory efficient.
Characteristic 6 Speed of execution:	Code is slower to execute.	Code is faster to execute.

J277 - 2.5 Programming languages and IDEs

Characteristics and purpose of different levels of programming language – LOW LEVEL LANGUAGE

100100
10000000011
101010101010
0000000001111

Low level language



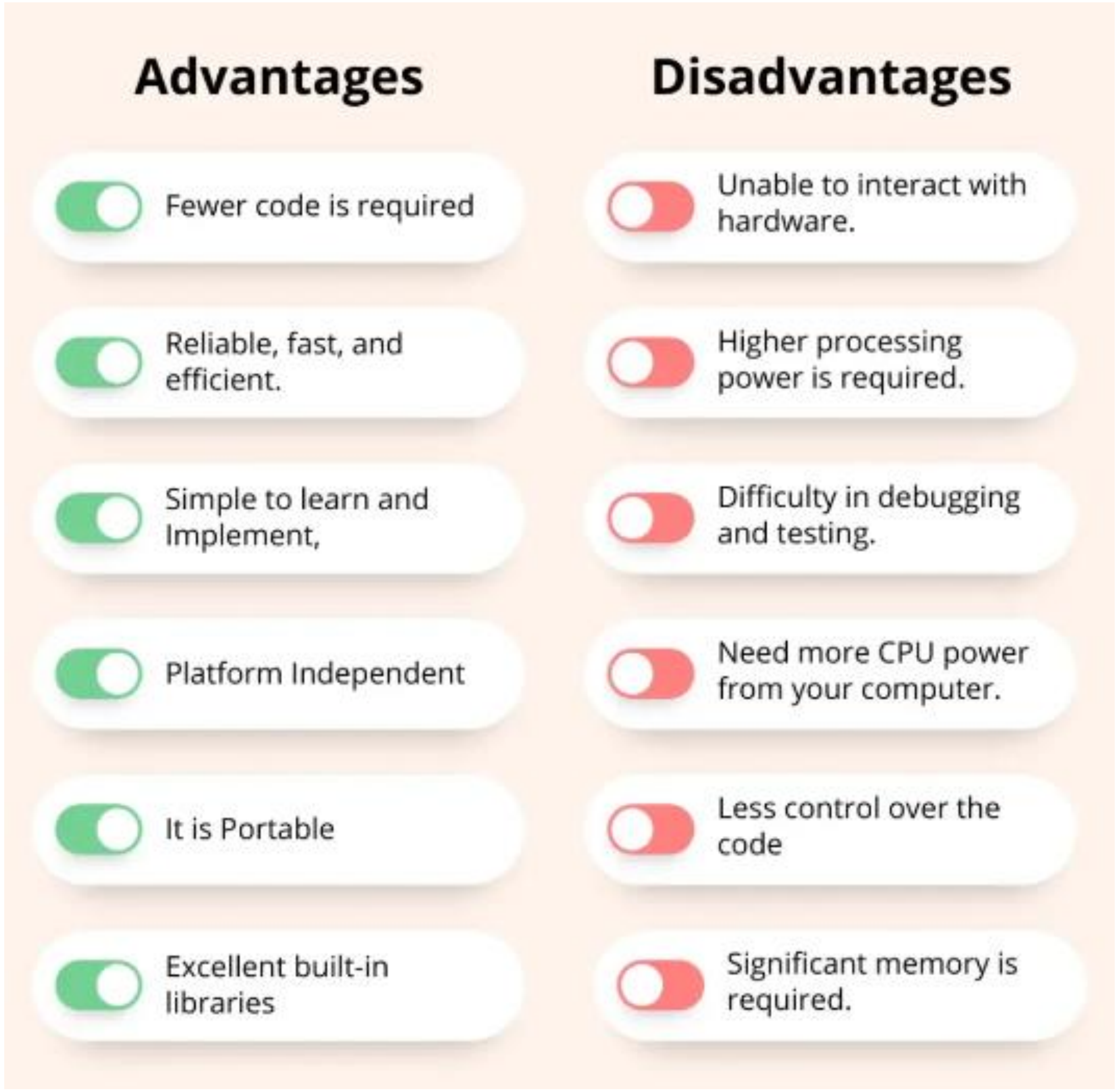
ADVANTAGES		DISADVANTAGES
The errors and bugs in assembly language can be easily tracked and solved.	1	The syntax used in assembly language is complicated to learn.
The mistakes in assembly language are fewer compared to other languages.	2	Assembly language is complex to understand and execute.
Assembly language is fast and can implement programs faster than others.	3	Needs more memory to run extensive programs and codes.
Assembly language is not portable; therefore differently used in different computers.	4	The code written in assembly language is lengthy and takes more time and effort to write and compile code.
The programmer can remember the storage locations without remembering the storage locations.	5	It requires more significant memory to execute more extensive programs.
It is more accessible than machine language	6	It is easy in machine language but takes massive time for coding.
It can perform and compile more complex tasks.	7	It takes an enormous amount of time to code and resolves issues.

J277 - 2.5 Programming languages and IDEs

Characteristics and purpose of different levels of programming language – HIGH LEVEL LANGUAGE

```
if(i<5)
{
printf("I am true block ");
}
else{
printf("I am false block");
}
```

High level language



Advantages	Disadvantages
Fewer code is required	Unable to interact with hardware.
Reliable, fast, and efficient.	Higher processing power is required.
Simple to learn and Implement,	Difficulty in debugging and testing.
Platform Independent	Need more CPU power from your computer.
It is Portable	Less control over the code
Excellent built-in libraries	Significant memory is required.

J277 - 2.5 Programming languages and IDEs

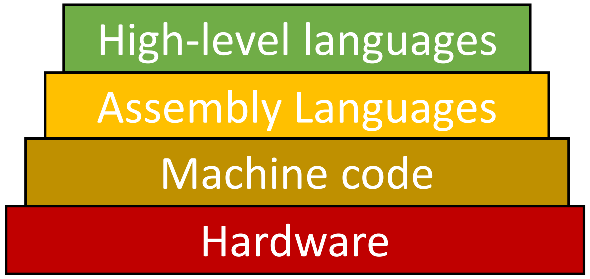
What is the relationship between machine code and assembly language?

Low-level language :	**Low-level languages** are languages that sit close to the computer's **instruction set**. An instruction set is the set of instructions that the processor understands. Two types of low-level language are: **Machine code & Assembly language.**
Machine code :	Machine code is the set of instructions that a CPU understands directly and can act upon. A program written in machine code would consist of only 0s and 1s - **binary**.

The difference between machine code and assembly languages:

- Assembly languages are written by programmers in assembly code.
- It is often used to develop software for embedded systems and controlling specific hardware components.
- They first have to be translated by an assembler into machine code before being run.
- Once run the actual CPU executes the machine code.
- Machine code is specific to each type of processor.
- The relationship between assembly code and machine code is one-to-one.
- This means that one line of assembly code translates into one specific line of binary machine code.

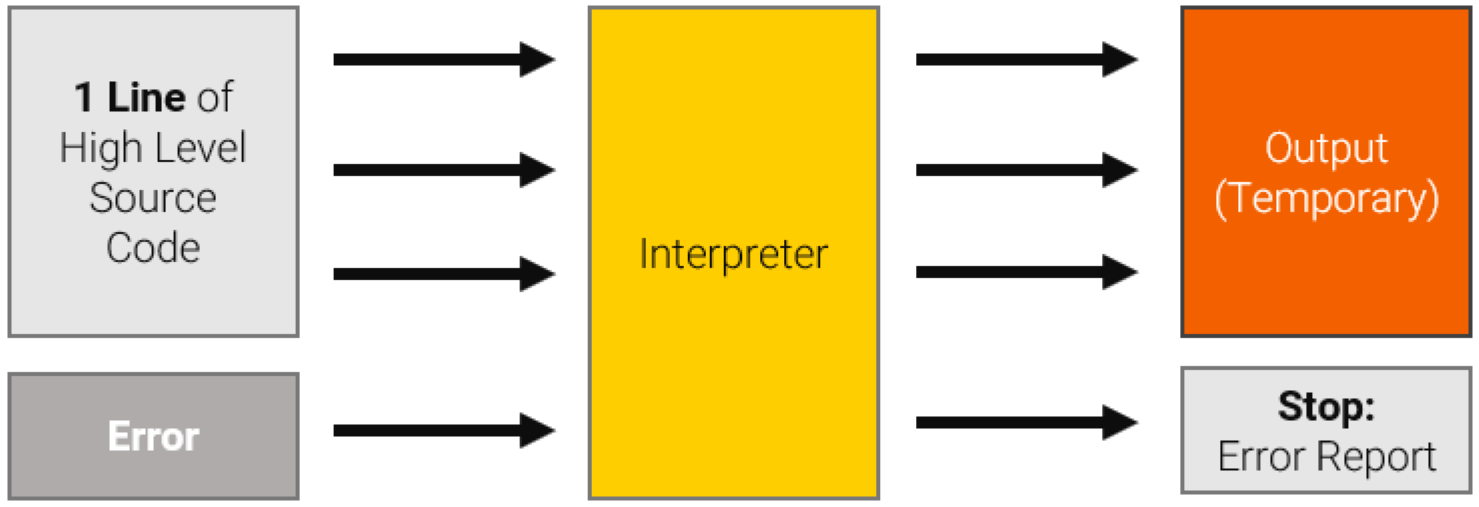
Hierarchy of languages:



J277 - 2.5 Programming languages and IDEs

The purpose of translators

Translators are needed to translate programs written in high level languages into the machine code that a computer understands. Tools exist to help programmers develop error-free code.

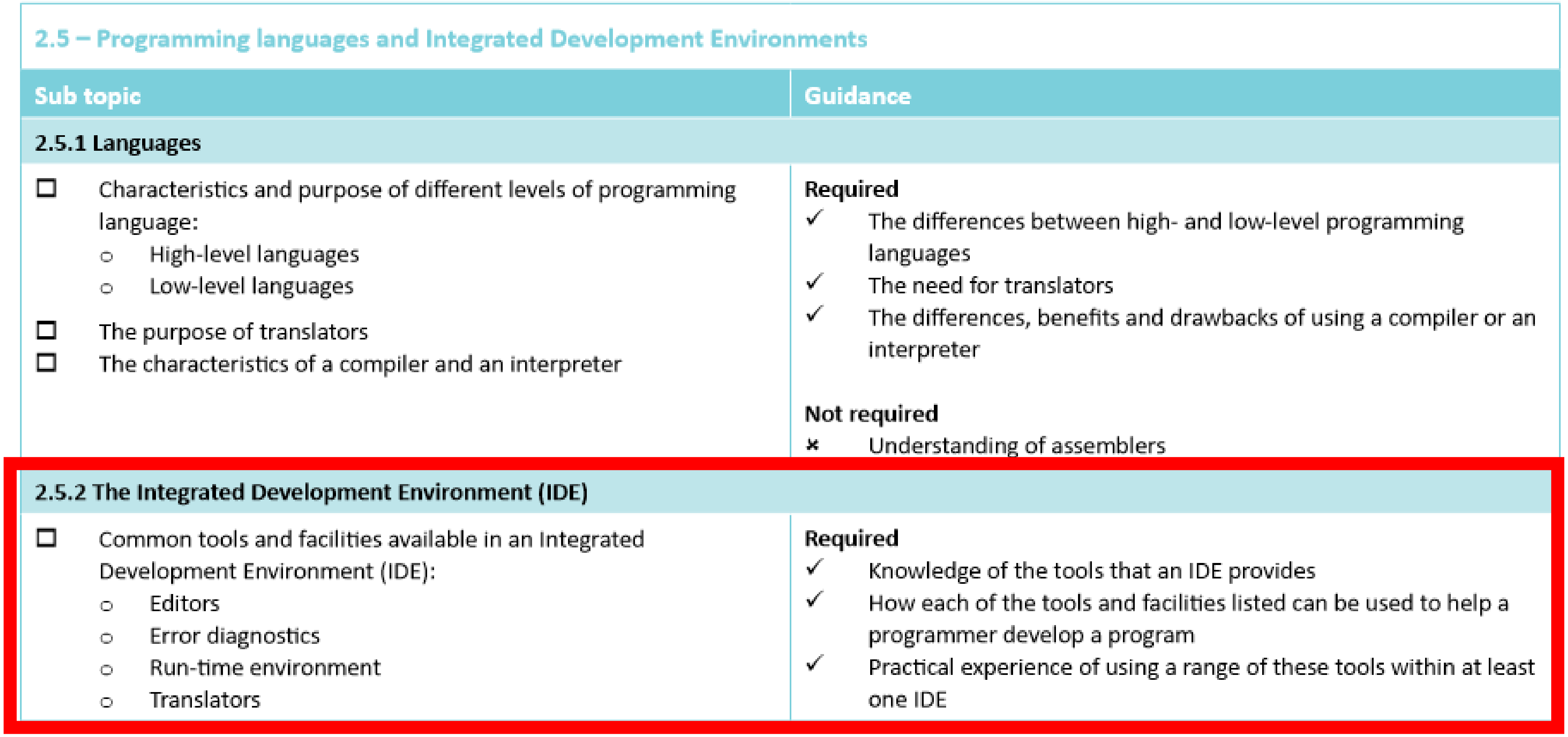


J277 - 2.5 Programming languages and IDEs

The characteristics of a compiler and an interpreter

	Compiler	Interpreter
Description:	Translates source code from a high level language into object code and then into machine code to be processed by the CPU.	Translates source code from a high level language into machine code to be processed by the CPU.
Feature:	The whole program is translated to machine code before it is run.	The program is translated line by line as the program is running.
Ease of writing code:	Easier to write code as it is close to English language, but the program will not run with syntax errors in the code.	Easy to write code as it is close to English language. The program will run and stop when it finds a syntax error.
Impact of changing code:	Needs to be recompiled.	Does not need be recompiled and it is easy to try out commands.
Designed for a specific type of processor:	Yes	No
Need for translation software at run-time:	No	Yes
Speed of code execution:	Quick	Slow
Optimised code:	Yes	No
Source code is kept secret:	Yes	No

J277 - 2.5 Programming languages and IDEs



2.5 – Programming languages and Integrated Development Environments

Sub topic	Guidance
2.5.1 Languages	
☐ Characteristics and purpose of different levels of programming language: ○ High-level languages ○ Low-level languages ☐ The purpose of translators ☐ The characteristics of a compiler and an interpreter	**Required** ✓ The differences between high- and low-level programming languages ✓ The need for translators ✓ The differences, benefits and drawbacks of using a compiler or an interpreter **Not required** ✗ Understanding of assemblers
2.5.2 The Integrated Development Environment (IDE)	
☐ Common tools and facilities available in an Integrated Development Environment (IDE): ○ Editors ○ Error diagnostics ○ Run-time environment ○ Translators	**Required** ✓ Knowledge of the tools that an IDE provides ✓ How each of the tools and facilities listed can be used to help a programmer develop a program ✓ Practical experience of using a range of these tools within at least one IDE

J277 - 2.5 Programming languages and IDEs

Common tools and facilities available in an integrated development environment (IDE)

Writing large programs can be a complex task. To help the programmer write clear, maintainable code, various tools exist.

Feature	Description
Text editor	Allows you to add and edit code as well as to insert comments.
Runtime environment	Runs your program by converting your source code into machine code in order for it to be executed by the CPU.
Syntax checking	Checks for any potential syntax errors in line with the rules of the language you are writing in. This helps to avoid common syntax errors appearing at the point when code is executed.
Keyword highlighting	Colour codes command words, variables, and data types to make your code more readable and easier to debug.

Feature	Description
Debugging tools	Tools that help you to detect and locate errors so you can fix them.
Break point	A debugging tool that enables you to stop the program execution at a specific point to enable you to see the values of the variables. Some IDEs also allow you to step through the code line by line to trace the values of the variables.
Memory inspector	Displays the contents of memory so that you can see how it is being used by the program in order to help debug problems such as memory leak.
Threading	Debugging tool that allows you to see the threads currently running. The inspector allows you to suspend, resume, and see the status of each thread being executed by your program.