



IGCSE 0478 Computer Science Resources

Prepared by the Aitchison College Academic Portal

Paper Pattern

Paper 1	1 hour 45 minutes
Computer Systems	50%
75 marks	
Short-answer and structured questions	
Questions will be based on Topics 1–6 of the subject content	
All questions are compulsory	
No calculators are permitted	
Externally assessed	

Paper 2	1 hour 45 minutes
Algorithms, Programming and Logic	50%
75 marks	
Short-answer and structured questions and a scenario-based question	
Questions will be based on Topics 7–10 of the subject content	
All questions are compulsory	
No calculators are permitted	
Externally assessed	

Assessment objectives

The assessment objectives (AOs) are:

AO1

Demonstrate knowledge and understanding of the principles and concepts of computer science.

AO2

Apply knowledge and understanding of the principles and concepts of computer science to a given context, including the analysis and design of computational or programming problems.

AO3

Provide solutions to problems by:

- evaluating computer systems
- making reasoned judgements
- presenting conclusions.

Weighting for assessment objectives

The approximate weightings allocated to each of the assessment objectives (AOs) are summarised below.

Assessment objectives as a percentage of the qualification

Assessment objective	Weighting in IGCSE %
AO1	40
AO2	40
AO3	20
Total	100

Assessment objectives as a percentage of each component

Assessment objective	Weighting in components %	
	Paper 1	Paper 2
AO1	60	20
AO2	20	60
AO3	20	20
Total	100	100



Computer Science - Paper 1

Table of Contents

Chapter 1: Number Systems and File Compression

Chapter 2: Data Transmission and Error Detection Methods

Chapter 3: Hardware

Chapter 4: Software

Chapter 5: The Internet and Its Uses

Chapter 6: Automated and Emerging Technologies

Chapter 1: Number Systems, Data Representation & File Management — Detailed Notes

1.1 Binary Systems

The binary system is the foundation of all computer operations. It is a base-2 number system meaning it only uses two digits: 0 and 1. Each binary digit is called a bit, the smallest unit of data. A byte is a group of 8 bits, and a nibble is 4 bits.

Why Binary?

Computers use binary because circuits can easily represent two states: ON (1) and OFF (0). All data and instructions in a computer are represented in binary.

Units of Data

Bit (b): single binary digit (0 or 1)

Nibble: 4 bits

Byte (B): 8 bits

Kilobyte (KB): 1024 bytes (2^{10})

Megabyte (MB): 1024 KB (2^{20})

Gigabyte (GB): 1024 MB (2^{30})

Terabyte (TB): 1024 GB (2^{40})

Conversions: Multiply or divide by 1024. Example: $2048 \text{ KB} \div 1024 = 2 \text{ MB}$.

Binary to Denary Conversion

Example: $10110_2 = (1 \times 16) + (0 \times 8) + (1 \times 4) + (1 \times 2) + (0 \times 1) = 22_{10}$

Denary to Binary Conversion

Method 1: Repeated Division by 2 $\rightarrow 22 = 10110_2$

Method 2: Subtracting Powers of 2 $\rightarrow 22 = 10110_2$

Binary Arithmetic

$0+0=0$, $0+1=1$, $1+0=1$, $1+1=10$ (carry 1)

Example: $1011 + 0111 = 10010_2$

Signed Binary (Two's Complement)

Used to represent negative numbers.

Steps: Write positive number $\rightarrow$ Invert bits $\rightarrow$ Add 1.

Example: -5 in 8 bits $\rightarrow 00000101 \rightarrow 11111010 + 1 \rightarrow 11111011$

1.2 Hexadecimal Systems

The hexadecimal system is base-16, using digits 0–9 and A–F. Each hex digit equals 4 binary digits (a nibble).

Example: $A3_{16} = 10100011_2 \mid 1101001101_2 = 34D_{16}$

Uses:

- HTML colour codes (#FF5733)
- MAC addresses
- Memory addressing in programming

1.3 Character Sets & File Sizes

Computers store text using character codes that map symbols to numbers.

ASCII

7-bit standard (128 characters) includes letters, digits, punctuation, control codes. Extended ASCII uses 8 bits (256 chars).

Unicode

Supports characters from all global languages. UTF-8, UTF-16, and UTF-32 are common forms.

- UTF-8 uses 1–4 bytes per character.

- UTF-16 uses mostly 2 bytes.

- UTF-32 uses 4 bytes.

ASCII is a subset of Unicode. Both assign numeric codes to characters, but Unicode is global.

File Size Calculation

File size (bits) = number of characters × bits per character.

Example: $2000 \times 8 = 16000$ bits = 2000 bytes = 1.95 KB.

1.4 Images, Sound & Data Storage

Images are grids of pixels. Each pixel stores its colour in binary according to colour depth (bits per pixel). Resolution = width × height.

File Size = (Resolution × Colour Depth) ÷ 8

Example: $800 \times 600 \times 24 = 11,520,000$ bits = 1.37 MB

Sound Representation

Sound is analogue and must be digitised by sampling.

- Sample rate: samples per second (Hz)

- Sample resolution / Bit depth: bits per sample

- Amplitude: height of wave, determines loudness

- Channels: mono (1), stereo (2)

File Size = Sample Rate × Bit Depth × Channels × Duration

Example: 44.1kHz, 16-bit, stereo, 10s = 14,112,000 bits ≈ 1.68 MB

1.5 Data Compression

Purpose: To reduce file size for faster transfer and storage.

Lossless Compression

No data lost; file can be perfectly restored.

Used for text, spreadsheets, software, PNG, ZIP, FLAC.

Run Length Encoding (RLE)

Replaces repeated data with count + value.

Example: AAAAAA → 5A

Used for text (repeated letters), images (pixels), and sound (silent samples).

Lossy Compression

Removes data permanently to reduce file size.

Used in JPEG, MP3, MP4.

Methods: cropping, reducing colour depth, reducing sample rate/bit depth.

Lossy is irreversible but achieves much smaller file sizes.

1.6 File Management

File management organizes and maintains files within storage systems.

- Operating System manages files and permissions.
- File naming includes extensions (.txt, .jpg, .docx).
- Hierarchical structure: root → folders → subfolders → files.
- Paths: absolute or relative.
- Metadata: file size, type, creation date, author, permissions.

Note: File management is no longer in Chapter 1, but covered in Chapter 4.

Key Terms Summary

Bit Depth / Sample Resolution – bits per sample.
Amplitude – wave height (volume).
Sample Rate – samples per second.
Resolution – pixels in image.
Colour Depth – bits per pixel.
RLE – lossless compression storing repeated values.
Lossy Compression – removes data, irreversible.
Lossless Compression – keeps all data, reversible.
ASCII – 7-bit character set.
Unicode – universal multi-language encoding.
UTF-8 – variable-length (1–4 bytes).

Exam Tips

- Show working for conversions.
- Always write correct units (bits, bytes, KB, MB).
- Know ASCII vs Unicode, Lossy vs Lossless.
- Practise file size calculations for text, images, and sound.
- Mention amplitude and sample resolution when describing sound.

Unit 2: Data Transmission

1. Data Packets

- When data is sent across networks (especially over the internet), it is split into packets/datagrams instead of being sent as one continuous stream.
- Reason for packetization:
 - Easier to control/manage.
 - Allows more efficient use of network resources.
 - Prevents a single transmission from monopolizing the channel.

Packet Structure

1. Header

- Contains essential control information.
- Includes:
 - Sender IP address (origin).
 - Receiver IP address (destination).
 - Packet/sequence number (used to reassemble in correct order).
 - Size of packet (helps allocate buffer space).
- Sometimes includes protocol type (e.g., TCP/UDP).

2. Payload

- The actual data to be delivered (part of the overall message).
- Could be text, image data, file segments, etc.

3. Trailer

- Marks the end of the packet.
- Contains error-checking codes (e.g., CRC – Cyclic Redundancy Check) to detect corruption.
- CRC works by counting 1-bits, appending result, and checking at receiver's end.

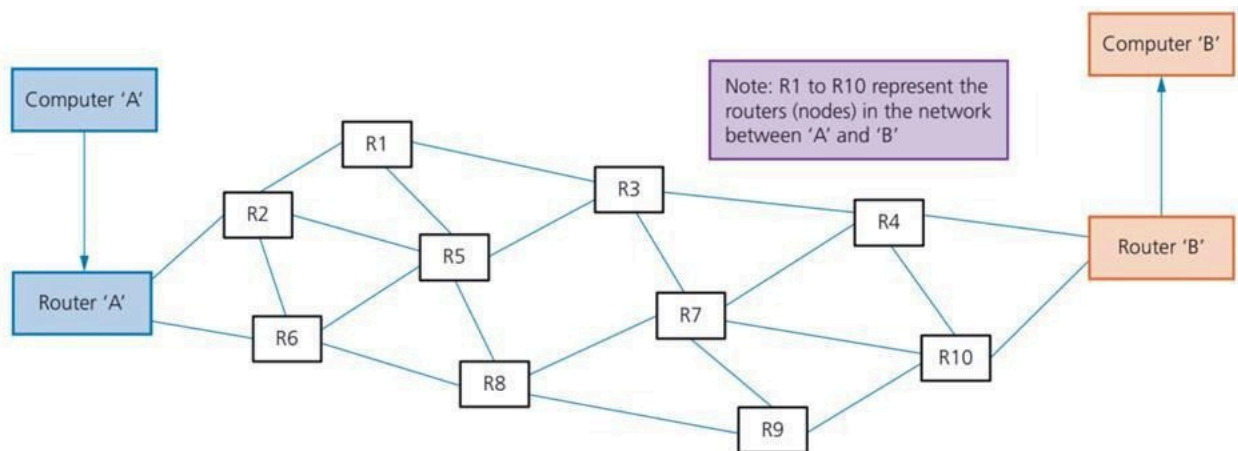
Key Features

- Packets may arrive out of order (different routes).
- Receiver reorders packets using sequence numbers.

- Hopping (packet passed node to node) prevents loss.
-

2. Packet Switching

- **Definition:** Method of transmission where data is divided into packets → each packet is transmitted independently via different routes → reassembled at destination.
- **Routers:** Devices at each node examine header information → choose next path.
- If the shortest route is busy, alternative free routes are chosen.



▲ Figure 2.2 Typical network showing possible routes between 'A' and 'B'

Advantages

1. No need for a single dedicated communication line.
2. Failed/busy routes can be avoided (fault tolerance).
3. Very scalable → easy to expand network usage.
4. High speeds possible, especially across global networks.

Disadvantages

1. Some packets may be lost in transmission.
 2. Real-time streaming suffers (errors, jitter, lag).
 3. Delays occur when packets are reassembled.
-

3. Transmission Modes

- **Communication protocols consider direction, method, and synchronization.**

Types of Direction

1. Simplex:

- **One-way only.**
- **Example: computer → printer.**

2. Half-Duplex:

- **Two-way but only one side at a time.**
- **Example: walkie-talkies.**

3. Full-Duplex:

- **Both directions simultaneously.**
 - **Example: broadband internet, phone calls.**
-

4. Methods of Data Transmission

Serial Transmission

- **Sends 1 bit at a time over a single channel/wire.**
- **Can be simplex, half-duplex, or full-duplex.**
- **Common in long-distance transmission (telephone lines, USB).**

Advantages:

- **Cheaper (fewer wires).**
- **Easier to synchronise (only one stream).**
- **Less interference (fewer channels).**
- **Reliable over long distances.**

Disadvantages:

- **Slower than parallel.**
 - **Usually for smaller data transfers.**
-

Parallel Transmission

- **Sends multiple bits simultaneously (e.g., 1 byte).**
- **Requires multiple wires/channels.**
- **Often used for short distances (e.g., internal components, printer ports).**

Advantages:

- **Faster than serial.**
- **Suitable for tasks needing high speed.**

Disadvantages:

- **Expensive (more hardware).**
 - **Data skewing/desynchronisation common over long distances.**
 - **More prone to interference.**
-

5. Universal Serial Bus (USB)

- **A standard serial data transmission method, widely used in computing.**
- **Uses four-wire shielded cable:**
 - **Red & black → power supply.**
 - **Green & white → data transmission.**

Features

- **Supports half-duplex and full-duplex.**
- **Devices are automatically detected and configured when plugged in.**
- **Computer loads drivers automatically (if missing, it downloads them).**

Advantages

- **Automatic device detection.**
- **Single orientation → prevents incorrect plugging.**
- **Industry standard (widely supported).**
- **Supports different data rates (1.5 Mbps – 5 Gbps).**
- **No external power supply needed.**
- **Error notifications provided.**
- **Supports hubs for multiple connections.**
- **Backward compatible with older versions.**

Disadvantages

- **Max cable length $\approx$ 5 metres.**
 - **Some older USB versions unsupported by new hardware.**
 - **Some USB versions relatively slow (<120 MB/s).**
-

6. Error Detection Methods

Parity Checks

- Adds a parity bit (extra bit at left).
- Even parity: total number of 1s must be even.
- Odd parity: total number of 1s must be odd.
- Sender and receiver agree on parity before transmission.
- Change in parity → error detected.

Limitations:

- Cannot detect if 2 bits flip.
- Can detect an error but not identify its position.

Parity Blocks:

- Apply parity to both rows and columns.
- Helps identify the exact bit where the error occurred.

Table 2.2

	parity bit	bit 2	bit 3	bit 4	bit 5	bit 6	bit 7	bit 8
byte 1	1	1	1	1	0	1	1	0
byte 2	1	0	0	1	0	1	0	1
byte 3	0	1	1	1	1	1	1	0
byte 4	1	0	0	0	0	0	1	0
byte 5	0	1	1	0	1	0	0	1
byte 6	1	0	0	0	1	0	0	0
byte 7	1	0	1	0	1	1	1	1
byte 8	0	0	0	1	1	0	1	0
byte 9	0	0	0	1	0	0	1	0
parity byte	1	1	0	1	0	0	0	1

A careful study of Table 2.2 shows the following:

- byte 8 (row 8) has incorrect parity (there are three 1-bits)
- bit 5 (column 5) also has incorrect parity (there are five 1-bits).

- Firstly, the table indicates that an error occurred during data transmission, as there is a change in the parity of one of the bytes.

· Secondly, the intersection of row 8 and column 5 identifies the position of the incorrect bit that caused the error. The 1-bit at this position should actually be a 0-bit. Therefore, byte 8 should correctly read as:

0 0 0 1 0 0 1 0

· This correction also restores even vertical parity in column 5 (which now contains four 1-bits). The byte can either be automatically corrected as shown above, or an error message may be sent back to the sender requesting re-transmission of the entire data block.

Checksum

- Sender calculates checksum using an agreed algorithm → attaches to data.
- Receiver recalculates checksum from received data.
- If mismatch → error → request resend.

Echo Check

- Data sent to receiver → receiver sends it back.
- Sender compares returned data with original.
- Limitation: Cannot tell if error occurred during sending or returning.

Check Digits

- Last digit in a code, calculated from the rest (using ISBN-13, Modulo-11 etc.).
- Used in barcodes, ISBNs, VINs.
- Detects human entry errors (e.g., mistyping, mis-scanning).

Modulo-11 Calculation Method:

Calculation 1 – Generation of the check digit from the other digits in a number

(In this example, we will assume the original number contained only 7 digits.)

The following algorithm generates the check digit from the other 7 digits:

- 1 each digit in the number is given a weighting of 8, 7, 6, 5, 4, 3 or 2 starting from the left (weightings start from 8 since the number will become eight-digit when the check digit is added)
- 2 the digit is multiplied by its weighting and then each value is added to make a total
- 3 the total is divided by 11
- 4 the remainder is then subtracted from 11 to find the check digit (note if the remainder is 10 then the check digit 'X' is used).

The example to be used has the following seven-digit number:

1 7-digit number: 4 1 5 6 7 1 0

weighting values: 8 7 6 5 4 3 2

2 sum: $(8 \times 4) + (7 \times 1) + (6 \times 5) + (5 \times 6) + (4 \times 7) + (3 \times 1) + (2 \times 0)$
 $= 32 + 7 + 30 + 30 + 28 + 3 + 0$
total = 130

3 divide total by 11: $130/11 = 11$ remainder 9

4 subtract remainder from 11: $11 - 9 = 2$ (check digit)

So we end up with the following eight-digit: 4 1 5 6 7 1 0 2

Calculation 2 – Re-calculation of the check digit from the eight-digit number (which now includes the check digit)

To check that the eight-digit number is correct, including its check digit, a similar process is followed:

- 1 each digit in the number is given a weighting of 8, 7, 6, 5, 4, 3, 2 or 1 starting from the left
- 2 the digit is multiplied by its weighting and then each value is added to make a total
- 3 the total is divided by 11
- 4 the number is correct if the remainder is zero

Using the 8-digit number: 4 1 5 6 7 1 0 2

1 weighting values: 8 7 6 5 4 3 2 1
2 sum: $(8 \times 4) + (7 \times 1) + (6 \times 5) + (5 \times 6) + (4 \times 7) + (3 \times 1) + (2 \times 0) + (1 \times 2)$
 $= 32 + 7 + 30 + 30 + 28 + 3 + 0 + 2$
total = 132

3 divide total by 11: $132/11 = 12$ remainder 0

4 remainder is 0, therefore number is correct

Automatic Repeat Requests (ARQ)

- Ensures reliable communication.
 - Uses:
 - Acknowledgements (ACK/NACK): Confirm receipt or request resend.
 - Timeouts: If no ACK in a set time, data resent.
 - Often used in mobile phone networks.
 - Can work with Cyclic Redundancy Check (CRC) , parity, or checksum for error detection.
-

7. Encryption

Purpose

- Prevents unauthorized access during transmission.
- Plaintext (readable) → encrypted → Ciphertext (unreadable).

Symmetric Encryption

- Same key used for encryption and decryption.
- Simple but less secure (key must be transmitted to receiver).

Asymmetric Encryption

- Uses two keys:
 - Public key (shared openly).
 - Private key (kept secret by receiver).
- Sender encrypts with public key → only receiver's private key can decrypt.
- More secure because private key is never transmitted.

Chapter # 3:Hardware

Computer Architecture

- *CPU*

- Central Processing Unit (CPU) is responsible for fetching, decoding and executing instructions.
- Laptops, Desktops and even mobile phones have a CPU.
- The user uses an input device to enter data and commands. The CPU processes the data by following instructions, and the results are shown through an output device.

INPUT → PROCESS → OUTPUT

<u>STEP</u>	<u>EXAMPLE</u>
Input	A mouse is used to click on a file icon.
Process	If the instruction being executed is OPEN, the system retrieves the file data and prepares it for viewing.
Output	The contents of the file are displayed on the screen for the user.

● **Microprocessor**

- A microprocessor is an integrated circuit that fits onto a single chip.
- It includes the CPU and may also contain additional components such as:
 - Memory controllers (handle data flow to and from memory)
 - Input / Output controllers

- Hence
 - CPU fetches, decodes and executes instructions.
 - Microprocessors are physical chips that contain CPU.
 - They are like Head (Microprocessor) and Brain (CPU).

➤ **Von Neumann Architecture**

- Von Neumann Architecture is a design of CPU proposed by Mathematician John Von Neumann.
- It describes how computer's memory, input/output devices, and processor function together.
- It is made up of:
 - **Control Unit (CU)**
 - Manages data flow within the CPU by sending control signals to direct its movement.
 - Interprets the instructions retrieved from memory.
 - **Arithmetic Logic Unit (ALU)**
 - Carries out arithmetic calculations.
 - Executes logical operations and decisions.
 - **Registers**
 - Registers are extremely small and incredibly fast memory units located within the CPU.
 - They temporarily store small amounts of data required during the fetch-execute cycle.

- Each register serves a specific function, and there are five main types.

<u>Name</u>	<u>Acronym</u>	<u>Purpose</u>
Program Counter	PC	Keeps track of address of next instruction to be executed, increasing by 1 with each cycle of the fetch-decode-execute process.
Memory Address Register	MAR	Holds the address in memory from which data or instruction will be fetched.
Memory Data Register	MDR	Stores the data or instruction that has just been fetched from memory.
Current Instruction Register	CIR	Stores the current instruction being decoded and executed by the CPU.
Accumulator	ACC	Stores the result of calculations performed by the Arithmetic Logic Unit (ALU).

○ Buses

- A bus is a set of parallel wires through which data / signals are transmitted from one component to another.
- There are 3 types of buses:

- **Address Bus** – One-way; it transfers address information, showing where data should be read from or written to.
- **Data Bus** – Two-way; it carries data or instructions to and from components.
- **Control Bus** – Two-way; it sends control signals and commands to tell components when to carry out read or write operations, etc.

➤ **Fetch-Decode-Execute Cycle**

- The Fetch-Decode-Execute Cycle is the process CPU performs billions of times each second to operate a computer.
- A computer receives input, processes it, and produces an output for the user.

Input: Pressing a key on the keyboard

Process: The CPU processes the keypress by referencing the program's instructions

Output: A letter appears on the screen in a text editor

● **Fetch Stage**

- During the fetch stage of the cycle, the program counter (PC) contains the address of the next instruction to be

retrieved from memory.

- This address is transferred to the memory address register (MAR).
- The address is then transmitted via the address bus, where it waits for a control signal to proceed.
- A signal is issued along the control bus by the control unit (CU) to communicate with main memory.
- Once the signal is received, the corresponding instruction or data is retrieved from memory and sent to the memory data register (MDR) through the data bus.
- A duplicate of the fetched instruction or data is then stored in the current instruction register (CIR) for processing.
- Finally, the program counter (PC) is incremented by 1, so it points to the subsequent instruction to be fetched in the next cycle.

● DecodeStage

- During the decode stage of the cycle, the CPU interprets the fetched instruction to determine what action is required.
 - This involves dividing the instruction into two main components:
 - Opcode – specifies the operation to be performed
 - Operand – indicates the data to be used or the memory location where the data can be found
- The operand can represent either a direct piece of data or an address

● Execute Stage

- During the execute stage, the CPU performs the action

specified by the decoded instruction.

- Examples of execute stage are:
 - Carrying out arithmetic or logical calculations
 - Storing a result or a piece of data back into main memory
 - Accessing a different memory location to retrieve required data

➤ **Characteristics of CPU**

- CPU has 3 common characteristics
 - Clock Speed
 - Cache size
 - Number of cores
- These have a significant impact on CPU's performance
 - **Clock Speed**
 - Clock speed is expressed in units called Hertz (Hz).
 - It indicates how many fetch-decode-execute cycles the CPU can perform in one second.
 - A higher clock speed allows more instructions to be fetched and executed every second.
 - Today's computers typically operate at speeds measured in Gigahertz (GHz), which represents billions of cycles per second.
 - For example, a 2.5GHz processor can handle as many as 2.5 billion instructions each second.

- **Cache Size**

- Cache is a small, fast memory located on or near the CPU.
- It temporarily stores copies of frequently used data and instructions for quick access.
- Larger cache size allows more commonly used data to be stored.
- This reduces the number of fetches from RAM, improving CPU performance.
- Cache is much faster than RAM, making data retrieval quicker.

- **Number of Cores**

- Each core functions like an individual CPU.
- Multi-core processors have multiple units that can process instructions simultaneously.
- A dual-core processor has 2 cores, each with its own CU, ALU, ACC, and registers.
- More cores allow smoother performance with demanding programs.
- Multiple cores boost CPU performance alongside clock speed.
- Example: A quad-core CPU at 3GHz can handle up to 12 billion instructions per second (4 cores $\times$ 3GHz).

➤ **CPU Instructions Set**

- An instruction set is a list of all commands a CPU can process.
- Each command is represented by a binary code, known as machine code.

➡ **Opcode & Operand**

- The opcode specifies the operation to perform.

- The operand indicates where the operation should be carried out.
- Instruction sets include mnemonics and their binary (machine code) equivalents.
- Once decoded, the CPU matches the opcode to the instruction set.
- This tells the CPU what action to take during execution.
- Instruction lists are machine specific
- Programs made for one instruction set won't run on a different type.

➤ **Embedded Systems**

- An embedded system is a computer designed to perform a specific task within a larger device.
- **Examples of embedded systems include:**
 - Hospital equipment, washing machines, coffee machines, satellite navigation systems, factory equipment, security systems, traffic lights.
- Characteristics of embedded system are
 - They are small
 - They use less power
 - They have lower cost

INPUT AND OUTPUT DEVICES

➤ **Input Devices**

- Input devices are hardware that let users send data or commands to a computer for processing.

<u>Device</u>	<u>Function</u>	<u>Where is it used</u>	<u>Why is it used</u>
<i>Barcode Scanner</i>	Scans barcodes to collect product information.	Supermarket Warehouses	Efficiently retrieves product information for inventory and checkout.
<i>Digital Camera</i>	Records photos or videos as digital files.	Photography Smartphones	High-quality visual content
<i>Keyboard</i>	Allows text and commands to be entered through key presses.	Computers Tablets Laptops	Type documents Enter data Control software
<i>Microphone</i>	Converts analog sound into digital signals by capturing audio.	Recording studios Smartphones Computers	Audio recording Voice communication
<i>Optical Mouse</i>	Controls the on-screen cursor by tracking hand movements across a surface.	Computers Laptops	Precise control and navigation on a computer screen
<i>QR Code Scanner</i>	Scans QR codes to retrieve stored information.	Retail shops Mobile apps	Quickly retrieves data from QR codes Website links or product information
<i>Touch Screen (Resistive)</i>	Senses touch through pressure on a flexible screen layer.	ATMs Older smartphones and tablets	Direct interaction with the device
<i>Touch Screen (Infrared)</i>	Registers touch by breaking infrared light beams across the screen.	Kiosks Large touch displays	Accurate touch detection Works well in various lighting conditions
<i>Touch Screen (Capacitive)</i>	Senses touch by detecting changes in electrical charge on the screen surface.	Modern smartphones Tablets Touchmonitors	Smooth, responsive touch multi-touch capabilities

<i>Two-Dimensional (2D) Scanner</i>	Captures flat, 2D images of documents or photos	Offices Libraries Schools	Digitises documents and photos for storage, editing, or sharing
<i>Three-Dimensional (3D) Scanner</i>	Captures detailed 3D shapes and dimensions of objects	Manufacturing Design Medical imaging	Creates digital models of objects for design, analysis, or reproduction

➤ **Output Devices**

- Output devices are hardware that receive data from a computer and present it to the user in a clear form. They allow the computer to display the results of processing.

<u><i>Name of device</i></u>	<u><i>What does it do?</i></u>	<u><i>Where is it used?</i></u>	<u><i>Why is it used?</i></u>
<i>Actuator</i>	Converts energy into motion to perform tasks as instructed by a computer or control system.	Make a robot's arm move Open a valve Rotate a motor	Necessary in any system that requires movement
<i>Digital Light Processing (DLP) projector</i>	Projects images or video onto a screen using light and tiny mirrors.	Cinemas Classrooms Large presentations	High-quality images
<i>Inkjet printer</i>	Prints text and images onto paper by spraying tiny droplets of ink	Home Schools Offices	Low cost compared to other types of printer

<i>Laser printer</i>	Heat from a laser fuses toner (fine powder) onto paper	Offices Schools	High-quality prints Fast print speed
<i>Light Emitting Diode (LED) screen</i>	Displays images and video using tiny lights called LEDs	Televisions Computer monitors Smartphones	High-quality images Bright Energy efficient Thin
<i>Liquid Crystal Display (LCD) projector</i>	Projects images or video onto a screen using light and liquid crystal panels.	Offices Classrooms Homecinemas	Bright Excellent balance of quality and affordability
<i>Liquid Crystal Display (LCD) screen</i>	Uses liquid crystals and a backlight to display images and video.	Televisions Monitors Smartphones Laptops	Energy efficient Thin Sharp images Lightweight
<i>Speaker</i>	Converts electrical signals into analogue sound waves to play music	Televisions Radios Smartphones	Multiple shapes and sizes
<i>3d printer</i>	Adds thin layers of plastic, one on top of another to form a three dimensional object	Manufacturing Medicine Education Home	Create prototypes Custom parts

➤ **Sensors**

- Sensors are input devices that measure physical properties like light, temperature etc. They are used in monitoring and control systems.

- Monitoring system tracks conditions, gathers data, and may issue warnings.
- Control system adjusts operations based on sensor inputs.
- Continuous process of using outputs as new inputs is called a feedback loop.

<u>Sensor type</u>	<u>What it measures</u>	<u>Typical use</u>
<i>Acoustic</i>	Sound levels	Detect changes in machinery sound levels Monitor noise pollution Identify suspicious sounds in security systems
<i>Accelerometer</i>	Acceleration rate, tilt, vibration	Detect sudden vehicle movement and activate safety features Sense device orientation in mobile phones
<i>Flow</i>	Rate of gas, liquid or powder flow	Detect changes in water flow through pipes
<i>Gas</i>	Presence of a gas e.g. carbon monoxide	Detect gas levels in confined spaces Monitor gas levels during leak repairs
<i>Humidity</i>	Levels of water vapour	Monitor humidity in greenhouses
<i>Infra-red</i>	Detecting motion or a heat source	Detect intruders breaking a security beam Measure heat radiation to locate people in emergencies
<i>Level</i>	Liquid levels	Detect petrol levels in a car tank Measure water levels in a tank Identify drops in water levels from leaks

<i>Light</i>	Light levels	Automatically switch on lights when it gets dark (e.g., streetlights, headlights)
<i>Magnetic field</i>	Presence and strength	Anti-lock braking systems (ABS) Monitor rotating machinery like turbines
<i>Moisture</i>	Presence and levels of moisture	Monitor soil moisture Detect dampness in buildings
<i>pH</i>	Acidity or alkaline	Monitor soil for optimal growing conditions Track pH levels in chemical processes
<i>Pressure</i>	Gas, liquid or physical pressure	Monitor tyre pressure Track pipe pressure in manufacturing
<i>Proximity</i>	Distance	Monitor object positions in robotics Prevent collisions in safety systems
<i>Temperature</i>	Temperature in Celsius, Fahrenheit or Kelvin	Maintain swimming pool temperature Control temperature in chemical processes

DATA STORAGE

➤ **Primary Storage**

- Primary storage is directly accessed by the CPU
- It is volatile (maintains data only when device is powered, except ROM)
- Small capacity
- Primary storage is much faster than secondary storage.

- Faster access speeds up operations like the Fetch-Execute Cycle.
- It holds data and instructions the CPU needs while the computer is on.
- Acts as short-term, working memory.
- Includes RAM, cache, and registers directly connected to the CPU.
- High performance makes it expensive, so capacity is limited.

➡ **RAM**

- Random Access Memory
- RAM is primary storage directly connected to CPU.
- Holds data and instructions currently in use.
- RAM is volatile, so data is lost when power is off.
- Data and instructions are copied from secondary storage for the CPU to use.
- RAM is very fast, much quicker than secondary storage.
- It is read/write, meaning data can be read and written.
- RAM has a much larger capacity than ROM.

➡ **ROM**

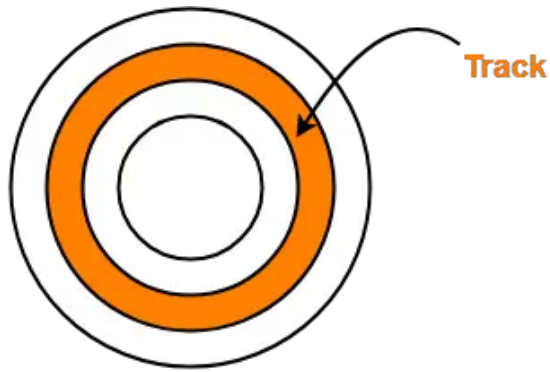
- ROM (Read Only Memory) is primary storage that stores startup instructions (Bootstrap). It contains the BIOS
- (Basic Input Output System). ROM is a small memory
- chip on the motherboard. It is fast—faster than
- secondary storage but slower than RAM. ROM is non-volatile, so data is kept when power is off. It is read-only,
- meaning data can only be read. ROM has a much smaller
- capacity than RAM.
-

➤ **Secondary Storage**

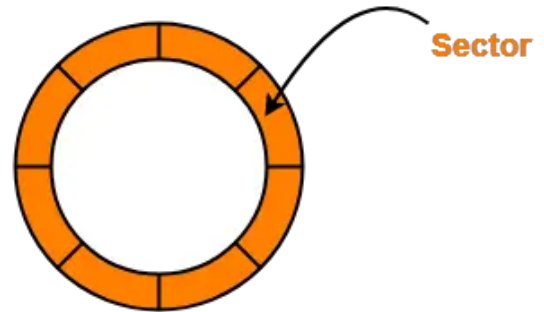
- Storage devices are **non-volatile secondary storage** that retain data on a computer.
- They store, access, and retrieve data such as software, documents, images, and videos.
- There are three types of storage:
 - Magnetic
 - Solid-state (flash memory)
 - Optical
- Secondary storage provides long-term storage for programs and data not in use.
- It is needed because ROM is read-only and RAM is volatile.
- Stores data and programs while the computer is off (non-volatile).
- Slower than primary storage but has much higher capacity.
- Ideal for backups and data archives.

➡ **Magnetic Storage**

- Magnetic storage is non-volatile media that uses magnetic polarity to store 0s and 1s.
- A magnetic hard disk contains several metal discs coated with magnetic material.
- These discs are called platters.
- Iron particles on the platters are magnetised to represent binary data (0 or 1).
- Each platter is divided into concentric circles (tracks) and wedge-shaped sectors.
- The point where they intersect is called a track sector.



Disk divided into tracks



Track divided into sectors

- The hard drive spins metal discs at high speed (typically 5400–7200 RPM) using a motor.
- A read/write arm, controlled by an actuator, moves the head to the data location.
- Data is read and written using electromagnets.

<i>Advantages</i>	<i>Disadvantages</i>
Capacity - High storage	Durability - Moving parts can get damaged if dropped
Cost - Low per gigabyte	Portability - Heavy & bulky making them less convenient for transport
Speed - Moderate read/write access	Reliability - Prone to mechanical failure
	Noise - Loud (spinning disks)

➡ **Solid State (Flash Memory)**

- Solid-state uses electronic circuits to store 0s and 1s.
- Eg Solid State Drives, USB
- Uses NAND and NOR gates to control the flow of electrons and store data persistently.
- Solid-state (flash memory) is made of tiny cells, each storing one bit (1 or 0).
- Each cell has a transistor that acts as an on/off switch.
- The transistor has two main parts:
 - Control gate: top layer; connects to the circuit and controls current flow.
 - Floating gate: holds a charge like a tiny battery, between two oxide insulating layers.
- To store data: a high voltage pushes electrons through the oxide onto the floating gate.
- To erase data: voltage is reversed, pulling electrons off the floating gate.

<i>Advantages</i>	<i>Disadvantages</i>
Capacity - Medium/high storage	Cost - Very high per gigabyte
Speed - Very fast read/write access	Reliability - Limited read/write cycles
Durability - No moving parts	
Portability - Small and no moving parts	

Noise - Silent

➡ **Optical Storage**

- Optical storage uses lasers to store data.
- The laser burns pits and lands on the disc's surface to represent binary 0s and 1s.
- Optical devices use a laser to read and write data.
- A moving arm directs the laser across the disk.
- In CD-Rs, the laser burns pits and lands to store data.
- The laser also reads data by detecting light reflections.
- Scattered light at pit-land changes is read by a sensor as binary data.

<i>Advantages</i>	<i>Disadvantages</i>
Cost - Very low per gigabyte	Capacity - Very low
Durability - No moving parts	Speed - Very slow read/write access
Portability - Small and no moving parts	Reliability - Prone to scratches
Noise - Silent	

➤ **Virtual Memory**

- Virtual memory is an extension of RAM located on secondary storage.

- Used when RAM is nearly full to keep the computer running.
 - Inactive programs and data are moved to virtual memory.
 - They are moved back to RAM when needed.
 - Data is stored in pages, which transfer between RAM and virtual memory as required.
- Much slower than RAM, so it reduces system performance.
- To avoid using it, increase RAM size.

➤ **Cloud Storage**

- Cloud storage is long-term (secondary) storage located remotely and accessed via the Internet.
- Data is stored on remote servers, usually magnetic (HDD) but increasingly solid-state (SSD).

<i>Advantages</i>	<i>Disadvantages</i>
Reliability - Access data anywhere, any time, from any device	Cost - Typically a monthly/yearly subscription plan
Security - Automatic backups	Availability - No access without an internet connection
Capacity - Endless storage space (cost permitting), uses less space on own device	
Collaboration - Multiple people can access the same file at the same time	

- Accessibility
 - Data can be accessed from anywhere with an internet connection.
 - Permission-based access allows easy file sharing and collaboration.
 - Works on any device connected to the internet.
 - Scalability
 - Storage capacity can be increased or decreased as needed.
 - Reliability
 - Providers use multiple servers and backups to prevent data loss.
 - Security
 - Offers data encryption and multi-factor authentication for protection.
 - No need to hire specialist IT staff—security is managed by the provider.
-
- No need to purchase local secondary storage devices.
 - One cloud storage center is more environmentally friendly than millions of individual servers.
 - Slow internet affects access.
 - Can be expensive for large data.
 - Subscription and data transfer costs.
 - Data may be intercepted.
 - Limited control over security

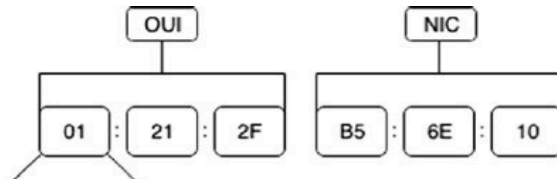
NETWORK HARDWARE

➤ **Network Interface Card**

- Network hardware enables devices to connect and communicate within a network.
- Common network hardware includes:
 - Router
 - Wireless Access Point (WAP)
 - Switch
 - Transmission media
 - Network Interface Card (NIC)
- **Network Interface Card (NIC)** allows a computer to connect to a network. Can be **wired or wireless**.
- Enables the computer to **send and receive data** over the network.

➡ **MAC Address**

- MAC (Media Access Control) address is a unique identifier for devices on a LAN.
- Assigned to a NIC at manufacture.
- Static: MAC addresses do not change.
- Allows switches to forward data to the correct device.
- Any device with a NIC has a MAC address.
- MAC address stays the same even if the device connects to a different network.
- MAC address is 12 hexadecimal digits (48 bits), usually in pairs.
- First three pairs = manufacturer ID (OUI).
- Last three pairs = NIC serial number.

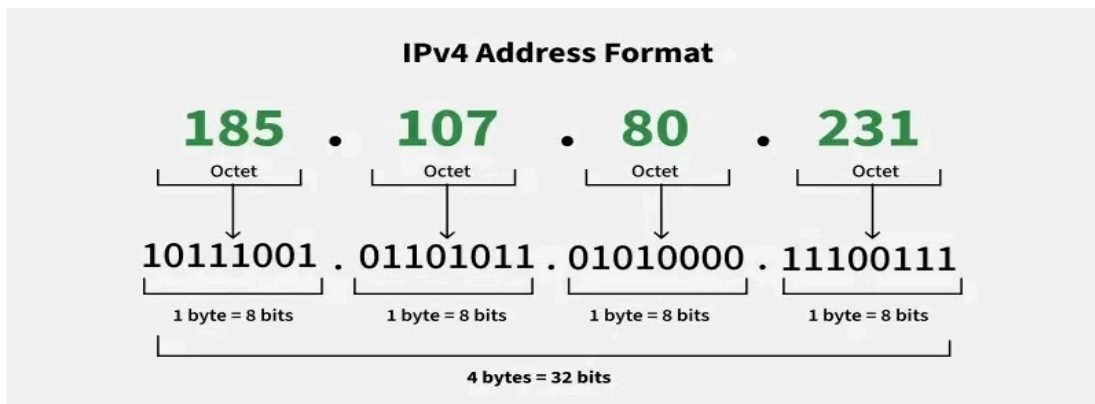


➡ IP Address

- IP(Internet Protocol) address is a unique identifier for devices on the Internet (WAN).
- Can be static (doesn't change) or dynamic (can change).
- Static IPs are used for servers and websites for consistent addressing.
- Dynamic IPs are assigned automatically by a DHCP server.
- IP addresses ensure data is delivered to the correct device.
- A device's IP address changes when it connects to a different network.

➡ IPv4

- IPv4 is represented as 4 blocks of decimal numbers (0–255) separated by full stops.
- Each block = 1 byte (8 bits), total address = 4 bytes (32 bits).



➡ IPv6

- IPv6 is represented as 8 blocks of 4 hexadecimal digits separated by colons.
- Each block = 2 bytes (16 bits), total address = 16 bytes (128 bits).



➤ Router

- Router directs data packets between different networks.
- Sends internet traffic to the correct devices in a home network.
- Connects LANs to the WAN (internet).
- Can manage and prioritise traffic to keep connections stable.
- Assigns IP addresses to devices on the network.

Example Question#1

Data is processed by the central processing unit (CPU) in a computer that has a Von Neumann architecture.

(a) Complete the table with the missing terms and definitions about processing data.

Term	Definition
current instruction register (CIR)	
.....	This is a processing unit within the CPU that can fetch, decode and execute instructions.
.....	This is a list of all the commands that can be processed by the CPU.
cache	
accumulator	

[5]

(b) Identify the component in the CPU that is responsible for decoding an instruction.

..... [1]

(c) Explain how the clock speed can affect the performance of a CPU.

.....

 [3]

Answer#1

(a)	<table><tr><th>Term</th><th>Definition</th></tr><tr><td>current instruction register (CIR)</td><td>stores the instruction that is being decoded/executed</td></tr><tr><td>core</td><td>This is a processing unit within the CPU that can fetch, decode and execute instructions.</td></tr><tr><td>instruction set</td><td>This is a list of all the commands that can be processed by the CPU.</td></tr><tr><td>cache</td><td>stores frequently used data/instructions</td></tr><tr><td>accumulator</td><td>stores the result of interim calculations</td></tr></table>	Term	Definition	current instruction register (CIR)	stores the instruction that is being decoded/executed	core	This is a processing unit within the CPU that can fetch, decode and execute instructions.	instruction set	This is a list of all the commands that can be processed by the CPU.	cache	stores frequently used data/instructions	accumulator	stores the result of interim calculations	5
Term	Definition													
current instruction register (CIR)	stores the instruction that is being decoded/executed													
core	This is a processing unit within the CPU that can fetch, decode and execute instructions.													
instruction set	This is a list of all the commands that can be processed by the CPU.													
cache	stores frequently used data/instructions													
accumulator	stores the result of interim calculations													
(b)	Control unit // CU	1												
(c)	Three from: - On each clock pulse, an FDE cycle is performed. - The clock speed is the number of FDE cycle performed in a second // The clock speed is the number of times the clock pulses per second/per unit of time. - If clock speed is increased more FDE cycles can be performed in a second // If clock speed is decreased fewer FDE cycles can be performed in a second. - If clock speed is increased, it can improve the performance of the CPU // If clock speed is decreases it decreases the performance of the CPU.	3												

EXAMPLE QUESTION#2

A weather station uses automated systems to collect and analyse data about the weather.

The automated system uses sensors.

(a) Give **one** other component that would be part of the automated system.

..... [1]

(b) One sensor that is part of the automated system is a temperature sensor.

Give **one** other sensor that could be part of the automated system.

..... [1]

(c) An alert is sent to an employee in the weather office if a temperature above 40 °C is detected.

(i) Explain how a temperature above 40 °C is detected for the alert.

.....
.....
.....
.....
.....
..... [3]

ANSWER # 2

(a)	Any one from: Examples: • Microprocessor • Actuator • Storage/memory	1
(b)	Any one from: Examples: • Humidity • Light • Level • Infrared	1
(c)(i)	Any three from: • A microprocessor is used. • Microprocessor received data from the sensor. • Microprocessor compares data against preset data of 40. • If the data is greater than 40 the microprocessor sends a signal to trigger the alert.	3

QUESTION # 3

A student's computer has primary storage. One example of primary storage is random access memory (RAM).

(a) Identify **one** other example of primary storage.

..... [1]

(b) Explain why RAM is an example of primary storage and **not** secondary storage.

.....

 [3]

ANSWER # 3

(a)	Any one from: • Read only memory // ROM • Cache	1
(b)	Any three from: • RAM is directly accessed by the CPU. • ... whereas secondary storage is not directly accessed by the CPU. • All secondary storage is non-volatile storage. • ... whereas RAM is volatile storage.	3

Chapter 4: Software— Detailed Notes

4.1 Types of Software

Software is a collection of instructions, data, or programs used to operate computers and perform specific tasks. There are two main types of software:

A. System Software

Definition: Software designed to operate the computer's hardware and provide a platform for application software to run. It manages hardware resources and controls system operations.

Examples: Operating Systems (Windows, macOS, Linux), Utility Programs (Antivirus, Disk Cleanup), Device Drivers.

B. Application Software

Definition: Software designed to perform specific tasks for the user — solving particular problems or carrying out a user-defined function.

Categories:

- **General-Purpose Software:** Used for various tasks (Word processors, Spreadsheets, Web browsers).
 - **Special-Purpose Software:** Performs one specific task (Payroll systems, Ticket booking).
 - **Bespoke / Custom-Written Software:** Developed for a specific organization's needs.
 - **Off-the-Shelf Software:** Ready-made, mass-produced for general users (Microsoft Office, Adobe Photoshop).
-

4.2 Operating Systems (OS)

An Operating System is the most essential system software. It acts as a bridge between the user/application software and computer hardware.

A. Key Functions of an Operating System

- **Memory Management:** Allocates and tracks RAM usage, ensuring programs don't interfere with one another. Uses virtual memory when RAM is full.
- **Processor / Task Management (Multitasking):** Allocates CPU time using scheduling methods (round-robin, priority-based) so multiple applications can run.

- **Input/Output and Peripheral Management:** Manages communication between CPU and devices (keyboard, printer, drives) via device drivers.
- **File Management:** Handles creation, naming, storage, retrieval, and deletion of files.
- **User Interface (HCI):** Provides interaction between user and system.
 - **GUI (Graphical User Interface):** Uses windows, icons, menus, and pointers (WIMP).
 - **CLI (Command Line Interface):** Text-based, requires command knowledge, less memory-intensive.
- **Security Management:** Controls authentication, passwords, and access rights.
- **Interrupt Handling:** Responds to signals requesting CPU attention.
- **Error Handling:** Detects and resolves hardware/software faults.

B. The Boot-Up Process

When powered on, the BIOS (Basic Input/Output System), stored in ROM, performs a POST (Power-On Self-Test). The BIOS then loads the Bootstrap Loader, which locates and loads the OS kernel into RAM. The OS initializes hardware and starts background processes.

C. Types of Operating Systems

- Single-User, Single-Tasking (MS-DOS)
- Single-User, Multitasking (Windows 11, macOS)
- Multi-User (UNIX)
- Real-Time OS (RTOS): Used in robotics, aircraft control.
- Distributed OS: Manages multiple networked systems.
- Embedded OS: Compact, efficient systems (ATMs, washing machines).

4.3 Utility Software

Utility programs maintain, secure, and optimize computer performance. They usually run within or alongside the OS.

Examples:

Utility Program	Function / Purpose
Antivirus Software	Detects, prevents, and removes malware. Must be updated regularly.
Disk Defragmenter	Reorganizes fragmented HDD files for faster access (not needed for SSDs).
File Compression Software	Reduces file size; may be lossless (reversible) or lossy (irreversible).
Backup Software	Creates copies of data on external drives or cloud for recovery.
	Removes temporary or duplicate files to free space.
System Cleanup / Disk Analysis	
Encryption Tools	Converts data into coded form to prevent unauthorized access.
Automatic Updates	Keeps OS and applications up to date with security patches.

4.4 Interrupts and Buffers

A. Interrupts

Definition: A signal sent to the CPU by hardware or software requesting immediate attention.

Purpose: Ensures urgent events (key press, power failure, device completion) are handled instantly.

Process:

1. Interrupt occurs (e.g., printer runs out of paper).
2. CPU pauses current process and saves its state.
3. Interrupt handler identifies priority.
4. CPU executes the Interrupt Service Routine (ISR).
5. CPU restores the saved state and resumes its task.

B. Buffers

Definition: Temporary RAM storage that holds data being transferred between devices operating at different speeds.

Purpose: Prevents faster components (CPU) from waiting on slower ones (printers, drives).

Example: A printer buffer stores data so the CPU can continue other operations while printing proceeds.

4.5 Device Drivers

Definition: Specialized programs that allow the OS to communicate with hardware devices.

Key Points:

- Translate OS instructions into device commands.
 - Essential for hardware functionality.
 - Updates improve performance and compatibility.
- Examples:** Printer drivers, GPU drivers, keyboard drivers.
-

4.6 Programming Languages and Translators

A. Types of Programming Languages

- **High-Level Languages (HLL):** Readable, portable, and easy to debug (Python, Java, C++).
- **Low-Level Languages (LLL):** Closer to hardware (Assembly, Machine Code).
Machine Code: Binary instructions executed directly by the CPU; machine-dependent.

B. Language Translators

Converts source code into machine code for execution.

Translator	Function	Output	Execution
Compiler	Translates the entire HLL program before execution.	Executable file.	Fast; errors reported after compilation.
Interpreter	Translates line-by-line and executes immediately.	None.	Slower; stops at first error.

Assembler	Converts assembly code to machine code.	Executable file.	Used in low-level/embedded systems.
------------------	---	------------------	-------------------------------------

Summary:

- Compiler → Fast execution, errors at the end.
- Interpreter → Easy debugging, slower runtime.
- Assembler → Used for direct hardware control.

4.7 Integrated Development Environments (IDEs)

Definition: A single software suite providing tools to write, test, and debug programs.

Common Features:

- **Code Editor:** Syntax highlighting, indentation, and line numbering.
- **Translator:** Built-in compiler/interpreter.
- **Debugger:** Step-through execution, breakpoints, variable inspection.
- **Auto-completion / Auto-correction:** Suggests and corrects code syntax.
- **Run-time Environment:** Executes and tests programs inside the IDE.
- **Output Console:** Displays program results or errors.
Examples: IDLE (Python), Eclipse (Java), Visual Studio (C#), Code::Blocks (C++).

4.8 System Software vs Application Software

Feature	System Software	Application Software
Purpose	Manages hardware & resources	Performs user tasks
Examples	OS, Drivers, Utilities	Word Processor, Browser

Runs	In background	Under user control
Dependenc y	Essential for operation	Optional, user-installed

4.9 System Software in Embedded Systems

Definition: Software built into specialized hardware for dedicated control.

Examples: Washing machines (cycle control), Printers (queue handling), Routers (packet routing).

Features: Small memory footprint, optimized for speed, real-time performance, minimal user modification.

4.10 Exam Tips

Understand System vs Application Software.

Know each OS function clearly.

Recall examples of utility software and translators.

Define interrupts and buffers with examples.

Identify IDE features and their uses.

Link embedded systems to real-time OS.

Ch5: The Internet and Its Uses

1. The Internet and the World Wide Web

Internet

- The Internet is a **global network of interconnected computers and devices** that communicate using standard transmission protocols.
- Uses **TCP/IP (Transmission Control Protocol / Internet Protocol)** for sending and receiving data between devices.
- Enables users worldwide to **exchange information and communicate** via:
 - o Email
 - o Voice and video calls
 - o Instant messaging
 - o File transfer and online collaboration
- It consists of millions of **private, public, academic, and government networks** linked together.
- The Internet acts as the **infrastructure**, whereas the World Wide Web is one of the **services** that run on it.

World Wide Web (WWW)

- The **World Wide Web** is a collection of **webpages, websites, and multimedia content** that can be accessed via the Internet.
 - It uses **HTTP or HTTPS protocols** for communication.
 - Webpages are usually written in **HTML (HyperText Markup Language)**.
 - Accessed through **web browsers** such as Chrome, Firefox, Edge, or Safari.
 - Each webpage has a **unique address (URL)** used to locate it on the Internet.
 - The WWW allows access to:
 - o Hyperlinked text and media
 - o Interactive web applications
 - o Databases and online services
-

2. Uniform Resource Locator (URL)

- A **URL (Uniform Resource Locator)** identifies the location of a specific webpage or file on the Internet.
- General format:
- protocol://website address/path/filename

Example: <https://www.notes.com/computer-science>

Main Components of a URL

1. **Protocol** – Defines the communication rules used (e.g., HTTP or HTTPS).
2. **Website Address (Domain Name)** – Divided into:
 - **Domain host:** Commonly www
 - **Domain name:** The name of the website (e.g., notes)
 - **Domain type / Top-Level Domain (TLD):**
 - Generic: .com, .org, .net, .gov
 - Country codes: .uk, .in, .pk, .cy, etc.
3. **Path** – Directory or file structure on the web server.
 - /computer-science represents a subfolder or file path.

3. HTTP and HTTPS

- **HTTP (Hypertext Transfer Protocol):** The protocol used for transferring and displaying webpages.
 - **HTTPS (Hypertext Transfer Protocol Secure):**
 - A more secure version of HTTP.
 - Encrypts data transmitted between browser and server using **SSL (Secure Socket Layer)** or **TLS (Transport Layer Security)**.
 - Ensures **data privacy and integrity**, preventing interception by attackers.
-

4. Web Browsers

- A **web browser** is a software application used to access and display webpages from the Internet.
 - Examples: Google Chrome, Mozilla Firefox, Microsoft Edge, Safari.
 - Main functions:
 - Connects to websites via HTTP(S) protocols.
 - **Translates HTML code** and displays formatted webpages.
 - Supports security through **SSL/TLS**.
 - Provides features such as bookmarks, browsing history, ad-blockers, and privacy settings.
-

5. Retrieval and Display of Webpages

1. The user enters a URL into the browser.
 2. The browser contacts a **DNS (Domain Name System) server** to translate the domain name into an **IP address**.
 3. The DNS returns the matching IP address (if it exists).
 4. The browser sends a request to that IP (web server).
 5. The web server responds by sending the requested webpage's HTML.
 6. The browser interprets and displays the webpage content to the user.
-

6. Cookies

- **Cookies** are small text files stored on the user's computer by websites.
- They store user preferences, login details, and browsing activity.
- Used for:
 - Auto-filling forms
 - Personalizing website experience

- Tracking activity for analytics or targeted advertising

Types of Cookies

a) Session Cookies

- Temporary; stored in **RAM** and deleted when the browser is closed.
- Do not store personal information.
- Example: Onlineshopping carts that hold items only during the session.

b) Persistent Cookies

- Stored on the **hard drive** until deleted or expired.
 - Used to remember login information and preferences.
-

Digital and Cryptocurrencies

1. Digital Currency

- **Digital currency** refers to electronic forms of money used for online payments.
 - Examples include:
 - Debit/Credit Cards
 - Mobile Payment Apps (PayPal, Apple Pay, Google Pay)
 - Bank transfers and digital wallets
 - It allows fast, convenient transactions but still relies on **centralized banking systems**.
-

2. Cryptocurrency

- Introduced to overcome problems in **centralized systems**.
- Uses **cryptography** for secure and verifiable transactions.
- Operates on a **decentralized Blockchain network**.
- Popular examples: Bitcoin, Ethereum, Litecoin.

- Advantages:
 - Enhanced security and privacy
 - No intermediary (bank) needed
 - Transactions recorded transparently on public ledgers
-

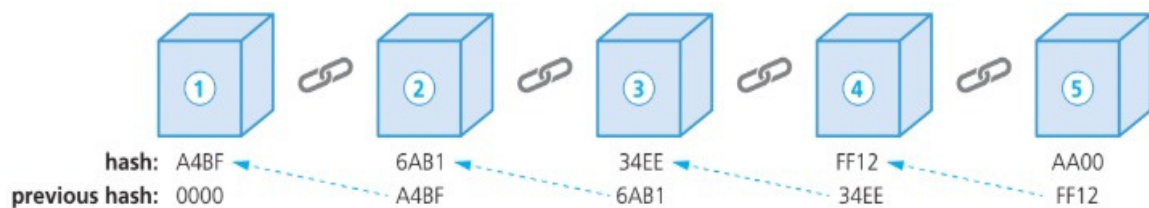
3. Blockchain Network

- A **blockchain** is a distributed database spread across multiple computers (nodes).
- Each transaction is grouped into a **block** and linked to the previous one, forming a chain.
- Impossible to alter data without the consent of the majority of network participants.

Structure of a Block

Each block contains:

- **Data:** Sender, receiver, transaction details, and amount.
- **Hash Value:** A unique identifier generated using a cryptographic algorithm.
- **Previous Hash Value:** Connects each block to the one before it.
- The **Genesis Block** is the first block with no previous hash (0000).



Cyber Security

Cybersecurity involves protecting computer systems, networks, and data from unauthorized access, attacks, and damage.

1. Common Types of Cyberattacks

Brute Force Attack

- The hacker repeatedly tries different password combinations until the correct one is found.
 - **Effects:** Access to confidential data such as emails or credit card details.
 - **Prevention:** Use long, complex passwords with numbers, symbols, and mixed case letters.
-

Data Interception

- Occurs when attackers intercept data being transmitted over a network.
 - Two common methods:
 - **Wardriving:** Illegally locating and using unsecured wireless networks.
 - **Packet Sniffing:** Capturing data packets sent over the network to extract information.
 - **Effects:** Datatheft, corruption, or system crashes.
 - **Prevention:**
 - Use **encryption** and **antivirus software**
 - Avoid untrusted software
 - Do not open suspicious emails or links
-

Distributed Denial of Service (DDoS) Attack

- Attackers overload a network or server by sending excessive traffic from multiple computers.
- **Effects:**
 - Makes websites or online services temporarily unavailable.
 - Prevents users from accessing their accounts or resources.
- **Prevention:**
 - Use firewalls, proxy servers, and intrusion detection systems.

Hacking

- Gaining **unauthorized access** to computer systems to steal, modify, or destroy data.
- **Effects:** Datatheft, identity fraud, or corruption.
- **Prevention:**
 - Firewalls and anti-hacking software
 - Strong passwords and user authentication
- **Hacking vs Cracking:**
 - **Hacking:** Gaining access to steal or manipulate data.
 - **Cracking:** Editing or modifying code maliciously (e.g., software cracking).

2. Malware (Malicious Software)

Types of Malware

- **Virus:** Self-replicating program that damages or deletes files.
- **Worm:** Replicates itself to infect entire networks.
- **Trojan Horse:** Disguised as legitimate software to trick users.
- **Ransomware:** Encrypts user data until ransom is paid.
- **Spyware:** Monitors user activity and sends information to attackers.
- **Adware:** Displays intrusive ads and can slow down system performance.

3. Phishing

- Fraudulent emails or websites designed to trick users into sharing sensitive data (passwords, banking info).
- **Prevention:**
 - Avoid clicking unknown links.
 - Use anti-phishing filters.

- o Keep browsers up to date.
 - o Block pop-ups.
-

4. Pharming

- Redirects users to fake websites by installing malicious code on their computers.
 - **Effects:** Theft of login details and personal data.
 - **Prevention:**
 - o Use antivirus software
 - o Check URL spelling and HTTPS padlock icon
 - o Avoid suspicious links
-

5. Social Engineering

- Manipulating individuals into revealing confidential information (e.g., fake phone calls claiming bank issues).
 - Relies on **psychological deception** rather than technical hacking.
-

Protecting Data from Threats

Access Levels

- Assigning different permissions to users based on their roles.
- Example: Only doctors can access patient medical records.

Antivirus Software

- Detects and removes malware.
- Must be updated regularly to detect new threats.

Authentication

- Confirms user identity through:
 - o Passwords and PINs

- o One-Time Passwords (OTPs)
 - o Biometric data (fingerprints, facial recognition, retina scans)
-

Biometric Methods

Method	Benefits	Drawbacks
Fingerprint Scan	Easy to use, low storage needed	Intrusive, fails if finger is dirty/damaged
Retina Scan	Extremely accurate, unique to each person	Intrusive, expensive, time-consuming
Face Recognition	Non-intrusive, fast, affordable	Affected by lighting, aging, or glasses
Voice Recognition	Quick, non-intrusive, inexpensive	Easily recorded, affected by illness

Two-Step Verification

- Requires two forms of identification (e.g., password + OTP or fingerprint).
- Adds an additional layer of security.

Automatic Software Updates

- Software patches fix vulnerabilities and enhance protection.

Spelling and Tone in Emails

- Scam emails often have poor grammar and urgency (“Your account will be closed!”).

Firewalls

- Hardware or software that **monitors and filters incoming/outgoing traffic**.

- Prevents unauthorized access to or from private networks.

Proxy Servers

- Act as intermediaries between the user and the web server.
- Functions:
 - o Hides user IP address
 - o Filters and controls web traffic
 - o Blocks certain websites
 - o Protects the main server from DDoS or hacking attacks

Privacy Settings

- Limit visibility of user data and control who can view or interact with profiles.

SSL (Secure Socket Layer)

- A security protocol for **encryptedcommunication** between a browser and server.
- Protects data transmitted online from being intercepted.

Past Paper Questions

0478/12/M/J/19/Q6

Priya creates a website to sell her old comic books and superhero figures. (a) She uses HTML to create her website. The HTML she produces has both structure and presentation. Explain what is meant by HTML structure and presentation. Include an example of each.

Structure.....

.....

Presentation.....

.....

(b) Priya uses cookies in her website.

Five statements are given about cookies.

Tick (✓) to show if the statement is **True** or **False**.

Statement	True (✓)	False (✓)
Cookies can be used to store a customer's credit card details		
Cookies can be used to track the items a customer has viewed on a website		
Cookies will corrupt the data on a customer's computer		
Cookies are downloaded onto a customer's computer		
Cookies can be deleted from a customer's computer		

(c) Priya stores her website on a webserver. To transmit the website data to the webserver she uses parallel duplex data transmission.

Describe how data is transmitted using parallel duplex data transmission.

.....

.....

.....

.....

.....

.....
.....
.....

[4]

(d) Priya has a URL for her website.

State what is meant by a URL.

.....
.....

[1]

(e) Priya is concerned about a denial of service attack (DoS) occurring on her webserver.

(i) Explain what is meant by a denial of service attack.

.....
.....
.....
.....
.....
.....
.....

[4]

(ii) Give one security device that can be used to help prevent a denial of service attack.

.....

[1]

0478/13/M/J/20/Q7

Edie uses a firewall to help prevent her children from accessing websites that she does not want them to see.

(a) Describe how the firewall helps prevent her children from accessing these websites.

.....
.....
.....
.....
.....

.....
.....
.....

[4]

(b) Edie is concerned that her children may download a virus when accessing websites. State what is meant by a virus and explain what could happen if a virus was downloaded.

.....
.....
.....
.....

[4]

(c) Edie explains to her children how to identify if a website is secure.

(i) Give two ways that her children can identify if a website is secure.

- 1
- 2

[2]

(ii) Describe how a browser checks that a website is secure.

.....
.....
.....
.....
.....
.....
.....
.....
..... [4]

Mark Scheme

0478/12/M/J/19/Q6

Question	Answer	Marks																		
6(a)	Structure – This is the layout of the web page – e.g. placing an image alongside some text // example of tag, such as <div> Presentation – This is the formatting/style of the web page – e.g. the colour that is applied to some text // example of tag, such as <font-color>	4																		
6(b)	1 mark per each correct row. <table border="1"> <thead> <tr> <th>Statement</th><th>True (✓)</th><th>False (✓)</th></tr> </thead> <tbody> <tr> <td>Cookies can be used to store a customer's credit card details</td><td>✓</td><td></td></tr> <tr> <td>Cookies can be used to track the items a customer has viewed on a website</td><td>✓</td><td></td></tr> <tr> <td>Cookies will corrupt the data on a customer's computer</td><td></td><td>✓</td></tr> <tr> <td>Cookies are downloaded onto a customer's computer</td><td>✓</td><td></td></tr> <tr> <td>Cookies can be deleted from a customer's computer</td><td>✓</td><td></td></tr> </tbody> </table>	Statement	True (✓)	False (✓)	Cookies can be used to store a customer's credit card details	✓		Cookies can be used to track the items a customer has viewed on a website	✓		Cookies will corrupt the data on a customer's computer		✓	Cookies are downloaded onto a customer's computer	✓		Cookies can be deleted from a customer's computer	✓		5
Statement	True (✓)	False (✓)																		
Cookies can be used to store a customer's credit card details	✓																			
Cookies can be used to track the items a customer has viewed on a website	✓																			
Cookies will corrupt the data on a customer's computer		✓																		
Cookies are downloaded onto a customer's computer	✓																			
Cookies can be deleted from a customer's computer	✓																			

Question	Answer	Marks
6(c)	– Several/multiple bits are transmitted at a time/simultaneously – Several/multiple wires are used – Data is transmitted in both directions ... – ... at the same time/simultaneously	4
6(d)	One from: – Uniform resource locator – The website's address – User friendly version of the IP address	1
6(e)(i)	Four from: – Designed to deny people access to a website – A large number/numerous requests are sent (to a server) ... – ... all at the same time – The server is unable to respond/struggles to respond to all the requests – The server fails/times out as a result	4
6(e)(ii)	One from: – Proxy server – Firewall	1

0478/13/M/J/20/Q7

Question	Answer	Marks
7(a)	Any four from: – Examines outgoing traffic to check what is being requested – Examines incoming traffic to check the content of what is being received – Sets rules/criteria for websites that can/cannot be accessed // creates a blacklist – Check if traffic meets/does not meet rules/criteria	4
7(b)	Any three from: – Software that can replicate itself – It could cause the computer to crash / run slow / generate errors – It could delete/damage files – It could fill up the storage space – It could stop the hardware being able to communicate – It could spread to other devices on the network	3

Question	Answer	Marks
7(c)(i)	Any two from: – Locked padlock – HTTPS – View the certificate	2
7(c)(ii)	Any four from: – requests web server to identify itself/view the (SSL) certificate – receives a copy of the (SSL) certificate, sent from the webserver – checks if (SSL) certificate is authentic/trustworthy – sends signal back to webserver that the certificate is authentic/trustworthy – starts to transmit data once connection is established as secure	4

CHAPTER 6 Automated System Notes

6.1 Introduction to Automated Systems

Definition:

An automated system performs tasks automatically with little or no human intervention. It uses computer control to monitor inputs and control outputs, often using feedback loops to maintain conditions.

Examples:

- Traffic light control systems
- Automatic washing machines
- Central heating and air conditioning systems
- Industrial robots and production lines
- Greenhouse environmental control

- Automatic doors and elevators

6.2 Components of an Automated System

Automated systems operate in three stages:

1. Input Devices (Sensors)

Function:

Sensors collect data from the environment and convert physical quantities (e.g. temperature, pressure, light) into electrical signals.

Common Sensors:

Sensor	Purpose	Example
Temperature sensor	Measures heat	Thermostat, oven
Light sensor (LDR)	Measures brightness	Streetlights
Pressure sensor	Detects pressure/force	Car tyres, elevators
Infrared sensor	Detects movement	Automatic doors
Sound sensor (microphone)	Detects sound	Voice-activated systems
Moisture/humidity sensor	Measures moisture	Greenhouses

pH sensor

Measures acidity/alkalinity

2. Processing (Microprocessor / Microcontroller)

Function:

Acts as the brain of the system, processing sensor data and making decisions using pre-programmed instructions.

Tasks:

- Compare input data with thresholds
- Send signals to actuators
- Store or log data
- Use feedback for adjustments

Example:

- If temperature $< 20^{\circ}\text{C}$ $\rightarrow$ heater ON
- If temperature $> 25^{\circ}\text{C}$ $\rightarrow$ heater OFF

3. Output Devices (Actuators)

Function:

Actuators carry out actions based on computer signals, converting electrical energy into movement or other effects.

Common Actuators:

Actuator

Function

Example

Motor	Produces rotational movement	Fans, robot arms
Heater	Produces heat	Ovens, incubators
Buzzer	Produces sound	Alarms
Light / LED	Provides illumination	Indicators
Pump	Moves liquids	Greenhouses, washing machines

6.3 The Control Process (Feedback Systems)

Open-Loop Control System

- No feedback used.
- System does not check if the desired result is achieved.
- Simple but less accurate.

Examples:

- Microwave timer turns off after a set time (not based on food temperature).
- Sprinkler turns on for 5 minutes regardless of soil moisture.

Flow:

Input → Processor → Output

Closed-Loop Control System (Feedback System)

- Uses feedback to monitor and adjust automatically.
- More accurate and self-correcting.

Example:

Air conditioner checks room temperature and adjusts cooling automatically.

Flow:

Input → Processor → Output → Feedback (sensor)

6.4 Examples of Automated Systems

1. Central Heating System

- Sensors: Temperature sensor
- Processor: Compares room temperature to target
- Actuators: Heater and fan

Control:

- If temperature < desired → heater ON
- If temperature > desired → heater OFF

2. Traffic Light Control System

- Sensors: Vehicle detectors (infrared/magnetic)
- Processor: Controls timing and sequence

- Actuators: Traffic lights

Features:

- Adjusts based on traffic flow
- Uses timed cycles or sensor data

3. Greenhouse Control System

- Sensors: Temperature, humidity, light
- Processor: Adjusts environmental conditions
- Actuators: Heaters, sprinklers, lamps

Control:

- If temperature too low → heater ON
- If light too low → lamp ON
- If humidity too low → water pump ON

4. Automatic Washing Machine

- Sensors: Temperature, water level, spin speed
- Processor: Controls washing, rinsing, and spinning
- Actuators: Valves, motors, heaters

Cycle Example:

1. Fill with water (until water-level sensor triggers)

2. Heat to required temperature
3. Wash → Rinse → Spin
4. Drain water and end cycle

5. Industrial Robots

- Sensors: Motion, position, pressure
- Processor: Controls robotic arms
- Actuators: Motors and grippers

Uses:

- Welding
- Painting
- Assembling components

6.5 Advantages and Disadvantages of Automated Systems

Advantages

- Operate continuously (24/7) without rest
- More accurate and consistent than humans
- Reduce human error
- Improve safety in dangerous environments
- Lower long-term costs

- Fast data processing and response time

Disadvantages

- Expensive initial setup
- Job loss for workers (unemployment risk)
- May malfunction or fail without notice
- Maintenance and repair require expertise
- Lack of human judgment or creativity

6.6 Summary

Automated systems combine sensors, processors, and actuators to perform tasks without human input.

They use feedback to maintain stable and accurate control, improving efficiency in industrial, household, and environmental systems.

Commonly Came Past Paper Questions – Chapter 6: Automated and Computer Control Systems

1. Component Identification

- Give one other component that would be part of the automated system.
- Give one other sensor that could be part of the automated system.
- Identify one output device that could be used in this automated system.

2. Sensor, Processor, and Actuator Operation

- Describe how a computer and sensors are used to monitor and control a greenhouse environment where the temperature must stay between 25°C and 30°C, and soil pH between 4 and 5.
- Describe how sensors and a microprocessor are used to control the temperature and pH in an industrial chemical process.
- Explain how a microprocessor uses input from sensors to control an automatic washing machine.
- Describe how a microprocessor would use data from sensors to control an air-conditioning system.

3. Naming and Applying Sensors or Actuators

- Name three different sensors and describe three applications where they are used.
- Name a suitable sensor for each of the following situations: detecting light level, measuring temperature, detecting movement, and measuring pressure.
- Explain the purpose of an actuator in an automated system and give one example of its use.

4. Advantages and Disadvantages of Automated Systems

- Describe the advantages and disadvantages of using automated systems instead of human workers.

- In a manufacturing company, what are the benefits and problems of using robots to assemble or paint products?
- State one disadvantage of using robots in place of human workers in factories.
- Explain why automated systems are more reliable than human-operated systems in some environments.

5. Monitoring vs Controlling

- Explain the difference between monitoring and control in a computer system.
- Give one example of a system that only monitors and one that both monitors and controls.
- Explain why feedback is necessary in a control system but not in a monitoring system.

6. Scenario-Based Questions

- A weather station uses automated systems to collect and analyse data about temperature, humidity, and wind speed.
 - (a) Give one other component that would be part of this system.
 - (b) Give one other sensor that could be part of this system.
 - (c) Explain how a temperature above 40°C would be detected and reported.
- A greenhouse needs to maintain constant humidity and temperature levels.
 - Describe how the sensors, microprocessor, and actuators work together to achieve this.
- A factory uses an automated robot to spray paint onto car parts.
 - Describe two problems the robot might face when painting.
 - Suggest how these problems could be overcome.

- An automatic street-lighting system switches lights on at sunset and off at sunrise.
 - Describe how sensors and a microprocessor control this process.
- An automatic security system uses an infrared sensor to detect movement.
 - Describe how this system operates when an intruder is detected.

Computer Science (Paper 2)

7 Algorithm Design and Problem Solving	2
7.1 Program Development Life Cycle	4
7.2 Computer Sub-Systems	5
7.2.1 Computer Sub-Systems Definitions	5
7.2.2 Advantages of Sub-Systems	6
7.3 Algorithms	6
7.3.1 Designing Algorithms	6
7.3.2 Explaining Algorithms	9
7.4 Standard Methods of Solution	9
7.5 Checks	13
7.5.1 Validation Checks	13
7.5.2 Verification Checks	14
7.6 Test Data	15
7.7 Identifying Errors	19
8 Programming Concepts	21
8.1 General Guidelines for Pseudocode	23
8.1.1 Data Types	24
8.1.2 Input and Output	25
8.1.2 Selection Statements	25
8.1.3 Iteration	27
8.1.4 Totalling and Counting Statements	29
8.1.5 String Handling	31
8.1.6 Arithmetic Operators	34
8.1.7 Relational Operators	34
8.1.8 Relational / Logical Operators	35
8.1.9 i) Procedures and Functions	35
8.1.9 ii) Library Routines	39
8.2 Arrays	41
8.2.1 Declaration	41
8.2.2 Use of Arrays	41

8.2.3 Writing and Reading	42
8.3 File Handling	42
8.3.1 Purpose of storing data in a file	43
8.3.2 File Commands	44
9 Databases	64
9.1 SQL (Structured Query Language)	66
10 Boolean Logic	73

7 Algorithm Design and Problem Solving

7 Algorithm design and problem-solving

Candidates should be able to:

- 1 Understand the program development life cycle, limited to: analysis, design, coding and testing

- 2
 - (a) Understand that every computer system is made up of sub-systems, which are made up of further sub-systems
 - (b) Understand how a problem can be decomposed into its component parts

 - (c) Use different methods to design and construct a solution to a problem

- 3 Explain the purpose of a given algorithm

Notes and guidance

- Including identifying each stage and performing these tasks for each stage:
 - analysis: abstraction, decomposition of the problem, identification of the problem and requirements
 - design: decomposition, structure diagrams, flowcharts, pseudocode
 - coding: writing program code and iterative testing
 - testing: testing program code with the use of test data

- Including:
 - inputs
 - processes
 - outputs
 - storage

- Including:
 - structure diagrams
 - flowcharts
 - pseudocode

- Including:
 - stating the purpose of an algorithm
 - describing the processes involved in an algorithm

Candidates should be able to:

4 Understand standard methods of solution

5 (a) Understand the need for validation checks to be made on input data and the different types of validation check

(b) Understand the need for verification checks to be made on input data and the different types of verification check

6 Suggest and apply suitable test data

7 Complete a trace table to document a dry-run of an algorithm

8 Identify errors in given algorithms and suggest ways of correcting these errors

9 Write and amend algorithms for given problems or scenarios, using: pseudocode, program code and flowcharts

Notes and guidance

- Limited to:
 - linear search
 - bubble sort
 - totalling
 - counting
 - finding maximum, minimum and average values
- Including:
 - range check
 - length check
 - type check
 - presence check
 - format check
 - check digit
- Including:
 - visual check
 - double entry check
- Limited to:
 - normal
 - abnormal
 - extreme
 - boundary
- Extreme data is the largest/smallest acceptable value
- Boundary data is the largest/smallest acceptable value and the corresponding smallest/largest rejected value
- Including, at each step in an algorithm:
 - variables
 - outputs
 - user prompts
- Precision is required when writing algorithms, e.g. $x > y$ is acceptable but x is greater than y is not acceptable

7.1 Program Development Life Cycle

There are four stages of the program development life cycle: analysis, design, coding, and testing.

Analysis

- Analysis is used to understand the problem the program intends to solve.
Analysis is further defined with the process of abstraction and decomposition.
Analysis removes unimportant sections to focus on the more important, refined elements such as functionality and requirements.
- Abstraction allows you to identify the fundamental components of the program, and allows you to create a document of requirements, simplifying the structure.
- Decomposition is the process of breaking down large problems into manageable sets of smaller problems.
- Benefits of decomposition are that smaller problems are easier to solve.
Moreover, each smaller problem can be solved and tested independently.

Design

- It involves using certain techniques to come up with a solution. Examples include Structure Diagrams, Flowcharts, and Pseudocode.

Coding

- Developers begin programming pieces of code (modules) in programming languages, to provide an overall solution. Iterative testing is conducted to ensure the modules interact accordingly with no errors.
- Some examples of high-level programming languages include Python Visual Basic, Java, C++, C#, etc.

Testing

- Once the overall program is structured, they are run many times with varying test data.
- This allows the program to work under specifications and reject invalid data.

7.2 Computer Sub-Systems

7.2.1 Computer Sub-Systems Definitions

- A smaller part of a system, working together with other sub-systems to create a fully functional computer system.
- For example, a car has 3 sub-system. They are:

Engine, Brakes, and Wheels

- Sub-systems can further have more sub-systems.
- Another example of computer sub-systems can be seen below:

CPU	Memory	Storage	Input devices	Output devices
Executes instructions	Stores data and instructions temporarily for the CPU	Stores data and software permanently	Allows a user to enter information	Displays information or creates a physical output

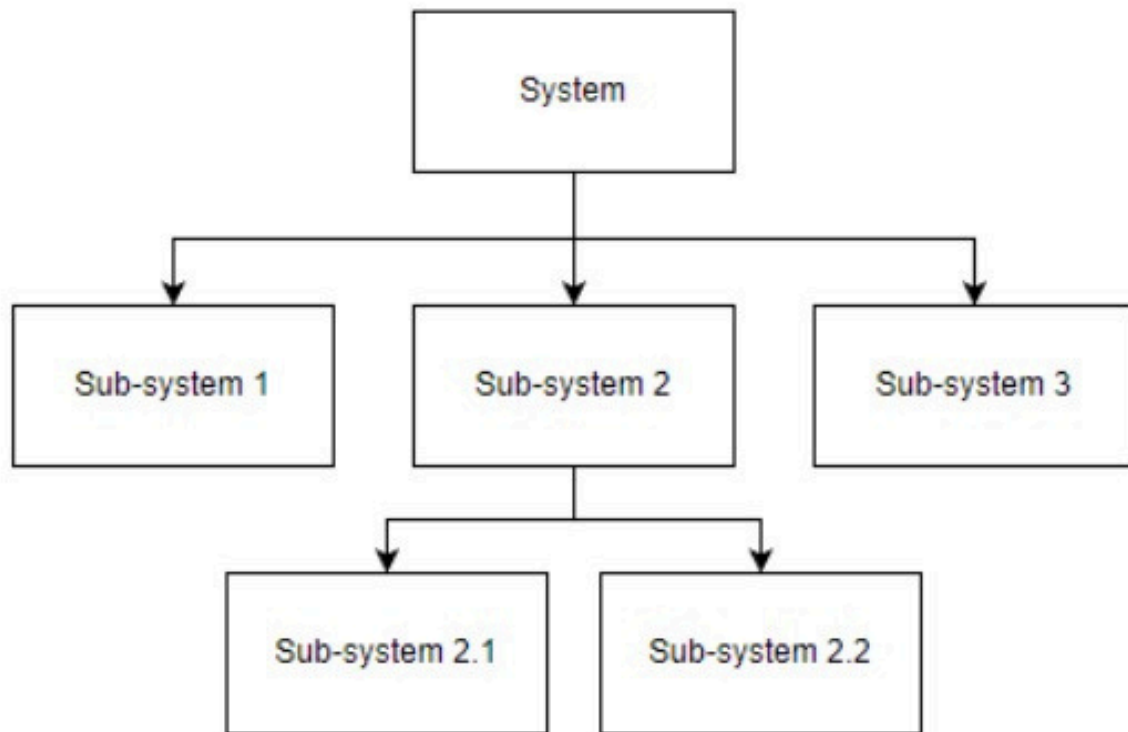
7.2.2 Advantages of Sub-Systems

- The ability to troubleshoot problems in a computer system is greatly enhanced by dividing it into sub-systems.
- Isolating each sub-system makes it easier to identify and fix issues, as every component can be examined separately without affecting the entire system. This approach is also essential in software development, where different sub-systems are used to ensure efficient operation and performance.
- It provides developers with a clear understanding of how these sub-systems work together to build and maintain complex systems.

7.3 Algorithms

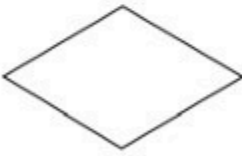
7.3.1 Designing Algorithms

- An algorithm is precise sets of rules to solve a specific problem. There are three main ways to design an algorithm:
 - ❖ Structure Diagrams
 - ❖ Flowcharts
 - ❖ Pseudocode
- A structure diagram shows us an hierarchical order of things, dividing problems into sub-sections. One such diagram can be observed below.



- Flowcharts are also visual tools, which can be used to display information as inputs, processes, and outputs. Lines are used to show flow of data.

Flowchart symbols

	Flow line	An arrow represents control passing between the connected shapes.
	Process	This shape represents something being performed or done.
	Subroutine	This shape represents a subroutine call that will relate to a separate, non-linked flowchart.
	Input/Output	This shape represents the input or output of something into or out of the flowchart.
	Decision	This shape represents a decision (Yes/No or True/False) that results in two lines representing the different possible outcomes.
	Terminator	This shape represents the 'Start' and 'Stop' of the process.

- Pseudocode is a text-based tool designed to describe an algorithm with ease. It provides more flexibility than programming languages.

(e.g.) $\text{DiceRoll} \leftarrow \text{ROUND}(\text{RANDOM}() * 6, 0) + 1$

7.3.2 Explaining Algorithms

- An algorithm should be well-designed to be fully interpreted by others. If a user does not know the purpose of an algorithm, it should use certain instructions to make its purpose clear. If the algorithm is complex to some extent, you should look at additional ways to understand it.

- Looking for comments.
- Considering the context.
- Testing the algorithm with different inputs.

(e.g.)

```
Count ← 1
Number ← 0
Total ← 0
REPEAT
    INPUT Number
    Total ← Total + Number
    Count ← Count + 1
UNTIL Count > 10
OUTPUT Total
```

- The purpose of the algorithm is to add ten user-entered numbers together and output the total. The processes are initializing three variables, inputting a user number, adding to two variables, repeating nine more times and outputting the final Total value.

7.4 Standard Methods of Solution

Linear Search:

- It is used to search for a certain value in a dataset.
- It checks all the values in the data structure being searched.

Q. Input a name and find out whether it is present in the 2D array *StudentName[1:50]*.

```
OUTPUT "Please enter the name you wish to find"
INPUT Name
Found ← FALSE
counter ← 1
REPEAT
    IF Name = StudentName[Counter]
        THEN
            Found ← TRUE
        ELSE
            counter ← counter + 1
        ENDIF
UNTIL Found OR counter > 50
IF Found
    THEN
        OUTPUT Name, " found at position ", counter, " in the list"
    ELSE
        OUTPUT "Name not found"
    ENDIF
```

Bubble Sort:

- Arranges items in a dataset in ascending or descending order.
- It swaps the place of 2 values if they do not follow the order.
- It requires a temporary variable to temporarily store values as they are being swapped.

Q. Arrange an array containing 10 student ages in ascending order.

```
Last ← 10
FOR Counter ← 1 TO 10
    OUTPUT "Enter student's age"
    INPUT Age
    StudentAge[Counter] ← Age
NEXT Counter
REPEAT
    Swap ← FALSE
    FOR Counter ← 1 TO Last - 1
        IF StudentAge[Counter] > StudentAge[Counter + 1]
            THEN
                Temp ← StudentAge[Counter]
                StudentAge[Counter] ← StudentAge[Counter + 1]
                StudentAge[Counter + 1] ← Temp
            ENDIF
    NEXT Counter
    Last ← Last - 1
UNTIL (NOT Swap) OR Last = 1
```

Totalling:

- The process of keeping a running total of values in the program.

```
Totalcost ← Totalcost + Foodcost
```

Counting:

- Number of times an action is performed in a program.

```
counter ← counter + 1
```

Maximum:

- Found by comparing values in a dataset to find the largest one.

```
Max ← 0
FOR counter ← 1 TO 10
  INPUT number
  IF number > Max THEN
    Max ← number
  ENDIF
NEXT counter
```

Minimum:

- Found by comparing values in a dataset to find the smallest one

```
Min ← 10000
FOR counter ← 1 TO 10
  INPUT number
  IF number < Min THEN
    Min ← number
  ENDIF
NEXT counter
```

Average:

- Found by dividing the total by the number of values.

```
FOR counter ← 1 TO 10
  INPUT number
  Total ← Total + number
NEXT counter
Average ← Total/10
```

7.5 Checks

7.5.1 Validation Checks

- Their function is to check that the data entered is reasonable.

Validation check	Function	Example
Range	The data is within a set range	REPEAT OUTPUT "Enter your marks" INPUT Marks UNTIL Marks > -1 AND Marks < 101
Length	The data is either an exact number of characters, or within a range of number of characters	REPEAT OUTPUT "Enter your password" INPUT Password UNTIL LENGTH(Password) >= 8
Type	The data conforms to the required data type	REPEAT OUTPUT "Enter your age" INPUT Age UNTIL DIV(Age, 1) = Age
Prescence	Data has been entered	REPEAT OUTPUT "Enter your name" INPUT Name UNTIL Name <> " "
Format	The data conforms to a specific format	REPEAT OUTPUT "Enter your device code" INPUT DeviceCode UNTIL SUBSTRING(DeviceCode, 1, 3) = "XYX" AND LENGTH(DeviceCode) = 6
Check digit	The final digit entered in a code is correct	REPEAT OUTPUT "Enter your device number" INPUT DeviceNumber OUTPUT "Enter your device code" INPUT DeviceCode UNTIL SUBSTRING(DeviceCode, 6, 1) = (2 * DeviceNumber)

Q. Write a program that outputs whether a person is allowed to drive the golf carts at a golf course based on if they are an adult or not. Include presence type and range checks.

4 A program needs to make sure the characters input for a product code meet these rules:

- The product code is six characters in length.
- The first two characters must be "PD".
- The last four characters must be a number in the range 1000 to 9999 inclusive.

4 A programmer is writing a data entry program for booking theatre seats.

The programmer needs the program to accept only whole numbers that are greater than or equal to one and less than or equal to six.

(a) Give the names of **two** validation checks that are required for this program.

1

2 [2]

(b) Complete this pseudocode to perform your **two** validation checks, using your answers given in (a):

OUTPUT "Please enter the number of seats you want to book "

INPUT Seats

7.5.2 Verification Checks

- To check that data has been accurately copied from one medium to another.

Verification Check	Function	Example
DoubleEntry	Requires data to be	REPEAT

	entered twice so that both entries can be compared to confirm that they are the same	<pre> OUTPUT "Enter your password" INPUT Password1 OUTPUT "Enter your password again" INPUT Password2 UNTIL Password1 = Password2 </pre>
Visual check	Manual check done by the user to see if the data entered is correct	Data is displayed on the screen and the user is asked whether it is correct.

7.6 Test Data

- Data that is used to test whether a program is working as designed or experiencing any errors.

Type of data	Description	Example (for a program that accepts 1-10 inclusive)
Normal	Data that will be accepted by the program	5
Abnormal	Data that should be rejected	138
Extreme	Largest/smallest acceptable value	1 or 10
Boundary	The largest/smallest acceptable value and the corresponding smallest/largest rejected value.	0,1 and 10,11

(b) Give the name of **one** verification check.

..... [1]

(c) Describe the difference between validation and verification.

.....

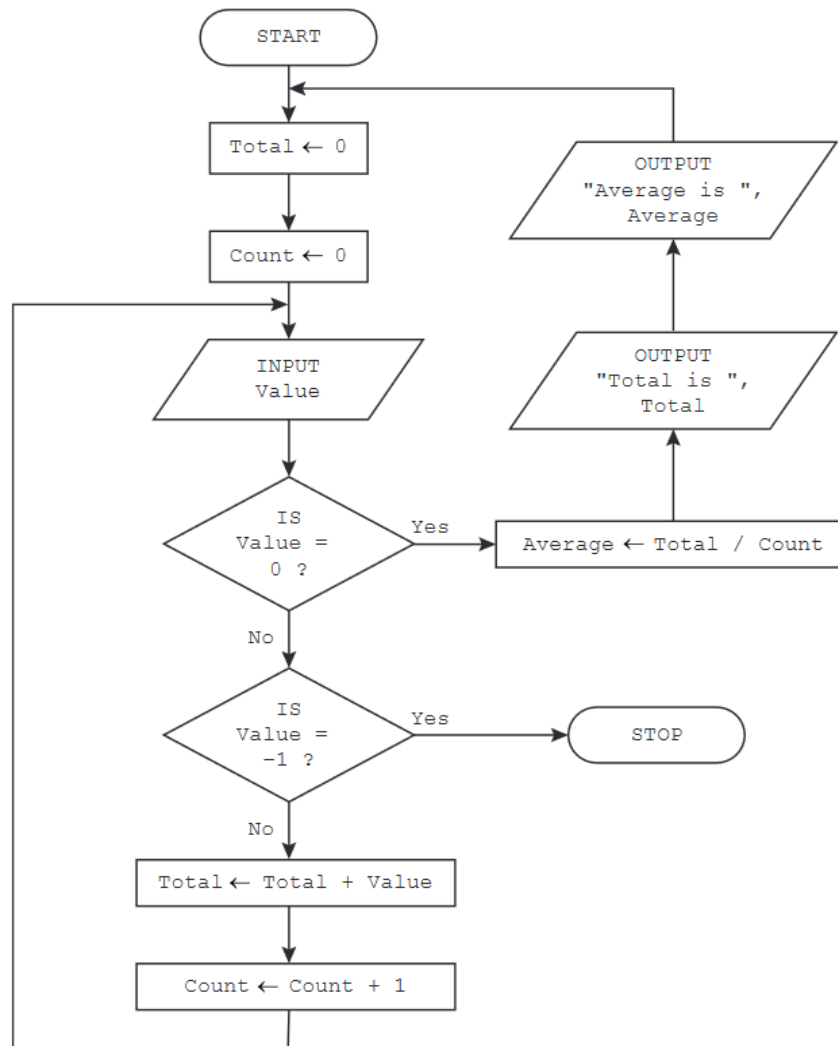
.....

.....

..... [2]

The flowchart represents an algorithm that performs a process on groups of values that are input. The algorithm will fail if the first value of any group is 0.

An input of -1 will terminate the algorithm.



Complete the trace table for the input data:

25, 35, 3, 0, 57, 20, 25, 18, 0, -1, 307, 40, 0

Value	Average	Total	Count	OUTPUT
		0	0	
25		25	1	
35		60	2	
3		63	3	
0	21			Total is 63
				Average is 21
		0	0	
57		57	1	
20		77	2	
25		102	3	
18		120	4	
0	30			Total is 120
				Average is 30
		0	0	
-1				

7.7 Identifying Errors

- 5 An algorithm has been written in pseudocode to calculate the profit when an item is sold. Values for cost price and selling price are input, the profit is calculated (selling price – cost price) and output. The input of zero for either value stops the algorithm.

```
01 REPEAT
02     OUTPUT "Enter cost price "
03     INPUT Cost
04     OUTPUT "Enter selling price "
05     OUTPUT Sell
06     IF Cost <> 0 OR Sell <> 0
07     THEN
08         Profit ← Sell - Cost
09         OUTPUT "Profit is ", Profit
10     NEXT
11 UNTIL Cost = 0 OR Sell = 0
```

- (a) Identify the line numbers of **three** errors in the pseudocode and suggest corrections.

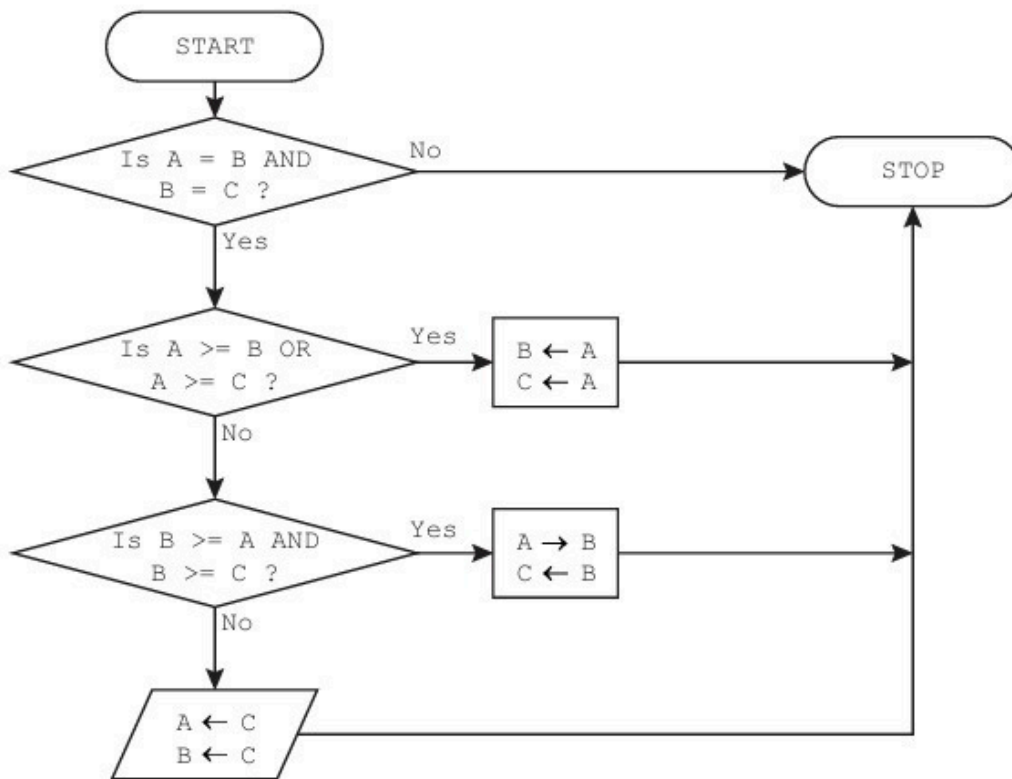
- Line 05 **OUTPUT** should be **INPUT**
- Line 06 **OR** should be **AND**
- Line 10 **NEXT** should be **ENDIF**

- 4 This pseudocode algorithm is intended to allow, at random, between 1 and 20 values to be entered and totalled. The total and average of the entered values are output at the end of the algorithm.

```
01 DECLARE Loop : STRING
02 DECLARE Limit : INTEGER
03 DECLARE Value : REAL
04 DECLARE Total : REAL
05 Total ← 0
06 Limit ← ROUND(RANDOM() * 19,0) + 1
07 IF Loop ← 1 TO Limit
08     OUTPUT "Enter a number"
09     INPUT Loop
10     Total ← Total * Value
11 NEXT Loop
12 OUTPUT "The total of the numbers entered is ", Total
13 OUTPUT "The average of the numbers entered is ", Total / Limit
```

- (a) Identify the line numbers of **four** errors in the pseudocode and suggest corrections.

- 9 A flowchart checks that values stored in three variables are identical. If they are different, the highest value is stored in all three variables.



Identify **four** errors in the flowchart and suggest a correction for each error. You may label each error on the diagram with the corresponding error number.

8 Programming Concepts

8.1 Programming concepts

Candidates should be able to:

- 1 Declare and use variables and constants
- 2 Understand and use basic data types
- 3 Understand and use input and output
- 4
 - (a) Understand and use the concept of sequence
 - (b) Understand and use the concept of selection
 - (c) Understand and use the concept of iteration
 - (d) Understand and use the concepts of totalling and counting
 - (e) Understand and use the concept of string handling
- 7 Complete a trace table to document a dry-run of an algorithm
- 8 Identify errors in given algorithms and suggest ways of correcting these errors
- 9 Write and amend algorithms for given problems or scenarios, using: pseudocode, program code and flowcharts

Notes and guidance

- Including:
 - integer
 - real
 - char
 - string
 - Boolean
- Including:
 - IF statements
 - CASE statements
- Including:
 - count-controlled loops
 - pre-condition loops
 - post-condition loops
- Including:
 - length
 - substring
 - upper
 - lower
- The first character of the string can be position zero or one
- Boundary data is the largest/smallest acceptable value and the corresponding smallest/largest rejected value
- Including, at each step in an algorithm:
 - variables
 - outputs
 - user prompts
- Precision is required when writing algorithms, e.g. $x > y$ is acceptable but x is greater than y is not acceptable
- See section 4 for flowchart symbols
- See section 4 for pseudocode

8.1 General Guidelines for Pseudocode

Variables:

An identifier whose value can change during the execution of a program.

Declaration	DECLARE <i><identifier></i> : <i><data_type></i>
Assignment	<i><identifier></i> ← <i><value></i>

Constants:

An identifier whose value cannot change during the execution of a program.

Declaration	CONSTANT <i><identifier></i> = <i><value></i>
-------------	--

Describe how variables and constants are used in programming.

variables / constants are used to store items of data

- the data stored in variables / constants are accessed by an identifier // named data stores
- the value of a variable may change during the execution of a program
- the value of a constant will remain the same during the execution of a program

Array:

Array is a linear data structure where all elements are arranged sequentially. It is a collection of elements of same data type stored at contiguous memory locations.

Declaration	CONSTANT <i><identifier></i> = <i><value></i>
-------------	--

Variable	Constant
CONSTANT Pi = 3.142	DECLARE amount : INTEGER

- Text should be evenly spaced with a consistent size throughout.
- Lines should be indented (usually by three spaces) to indicate that they are contained within a statement in a previous line.
- Keywords should be in uppercase, e.g. **REPEAT,ELSE**, etc.
- Comments should be included and are compulsory for the last question.
 - Comments are started by "//".
 - They are usually on an independent line.
 - Smaller comments may be on the same line as the code they refer to.
- Identifiers:
 - They should describe the variable, constant, procedure or function they represent.
 - They can be in mixed case
 - They should be considered case insensitive, i.e. you cannot create a different identifier by changing the case of the original identifier.
 - They can only contain letters (A–Z, a–z), digits (0–9) and the underscore character (_).
 - They must start with a letter and not a digit.

8.1.1 Data Types

Data Type	Explanation	Example
INTEGER	A whole number	67,526
REAL	A number containing a fractional part	42.6,-8.0
CHAR	A single character	'x','#'
STRING	A sequence of characters	"Hello world"
BOOLEAN	The values TRUE and FALSE	TRUE ; FALSE

0478/22 February/March 2023

- 9 A database table, `CapitalCity`, stores data about capital cities from around the world.

Field	Description
Identification	a unique identifier
Name	the name of the city
Country	the name of the country in which the city is located
Continent	the name of the continent in which the city is located
Population	the number of people living in the city
Founded	the year the city was founded
AverageIncome	average income of the city's population, in \$, rounded to 2 decimal places
Subway	whether or not the city has an underground railway system

- (a) Complete the table by writing **one** appropriate field from `CapitalCity` for each data type.

Each field must be different.

Data type	Field
Boolean	
integer	
real	
String	

[2]

0478/21May/June 2025

- 10 The variables `X`, `Y` and `Z` are used in a program: `X` stores a whole number, `Y` stores a decimal number and `Z` stores a flag that can be set to `TRUE` or `FALSE`

- (a) Write pseudocode statements to declare the variables `X`, `Y` and `Z`

0478/22 May/June 2025

- 9 The variables `A` and `B` are used in a program. `A` holds a whole number and `B` holds text.

- (a) Write pseudocode statements to declare the variables `A` and `B`

8.1.2 Input and Output

Input:

It is the value given to the computer by the user.

Output:

The information a computer displays, stores or transmits to another computer.

Input	INPUT <i><identifier></i>
Output	OUTPUT <i><identifier></i>

Sequence:

- It is the order in which the statements in a program are executed.
- A logical sequence is necessary for the program to function correctly.

8.1.2 Selection Statements

IF statements:

- If statements may or may not contain an **ELSE** clause.
- The else clause only functions when the condition for the if statement is false.
- Nested if statements are used to handle hierarchical decision making.

When there is no else clause	IF <i><condition></i> THEN <i><statement(s)></i> ENDIF
When there is an else clause	IF <i><condition></i> THEN <i><statement(s)></i> ELSE <i><statement(s)></i> ENDIF
Nested IF statement	IF <i><condition></i> THEN <i><statements></i> ELSE IF <i><condition></i> THEN <i><statements></i> ELSE <i><statements></i> ENDIF ENDIF

CASE statements:

- These statements allow one out of several branches of code to be executed, depending on the value of a variable.
- An **OTHERWISE** clause can be used as the last case if no cases apply.
- CASE statements are tested in order.
- When a case that applies is found, the other cases are not tested.

Without an otherwise clause	<pre>CASE OF <identifier> <value 1> : <statement1> <statement2> ... <value 2> : <statement1> <statement2> ENDCASE</pre>
With an otherwise clause	<pre>CASE OF <identifier> <value 1> : <statement1> <statement2> ... <value 2> : <statement1> <statement2> ... OTHERWISE : <statement1> <statement2> ... ENDCASE</pre>

0478/21 May/June 2024

- 5 Write the pseudocode statements to perform this task:
- accept the input of a whole number from 1 to 4 inclusive
 - use a CASE statement to:
 - output the number (1 to 4 inclusive) that was entered
 - output the word "ERROR" if a 1 to 4 inclusive number was **not** entered.

Q. Write a program to output the grade of a student according to the criteria:

- $X \geq 80 = A$ • 80
- $> x \geq 70 = B$ • 70
- $x \geq 60 = C$ • 60
- $> x \geq 50 = D$ • 50
- $> x < 50 = \text{FAIL}$

8.1.3 Iteration

Count-controlled loops:

- They are used when a known number of repetitions are required.
- In this loop, the counter is incremented automatically.

```
FOR <identifier> ← <value1> TO <value2>  
    <statement(s)>  
NEXT <identifier>
```

Pre-condition loops:

- It is used when the number of repetitions is not known.
- The condition is tested before the statements are executed.
- The statements may not be executed at all if the condition is evaluated as false.

```
WHILE <condition>  
    <statement(s)>  
ENDWHILE
```

Post-condition loops:

- It is also used when the number of repetitions is unknown.
- The condition is tested after the statements are executed once.
- The statements are always executed once.

```
REPEAT  
    <statement(s)>  
UNTIL <condition>
```

- 3 The following pseudocode algorithm uses a count-controlled loop to read in 10 names and store them in the array `Names[]`

```
FOR Count ← 1 TO 10  
    OUTPUT "Please enter a name"  
    INPUT Name  
    Names[Count] ← Name  
NEXT Count
```

Rewrite the algorithm in pseudocode, using a condition-controlled loop instead of a count-controlled loop.

With WHILE loop:

```
Count ← 1 //initialise counter variable  
WHILE Count <= 10 DO  
    OUTPUT "Please enter a name"
```

```

    INPUT Name Names[Count] ← Name //input name
    Count ← Count + 1
    //increment the counter
ENDWHILE

```

With REPEAT...UNTIL:

```

Count ← 1 //initialise counter variable
REPEAT
    OUTPUT "Please enter a name"
    INPUT Name
    Names[Count] ← Name
    Count ← Count + 1 //increment the counter
UNTIL Count > 10 //end the loop once it has repeated 10 times

```

04/8/22 February/March 2024

6 Describe **two** types of iteration that a programmer can use whilst writing a program.

04/8/22 February/March 2024

- (ii) Write the pseudocode for an algorithm to allow a number to be input into an appropriate variable and check whether the number is an integer. If the number is an integer, it is accepted. If the number is **not** an integer, an error message is output. Re-input of the number is required until a valid number is input.

8.1.4 Totalling and Counting Statements

Totalling:

- It is the process of keeping the total of values in a program.

Counting:

- It counts the number of times an action is performed in a program.

Q. Write an algorithm in pseudocode, using a single loop, to output the average of 50 numbers.

```

FOR counter ← 1 TO 50
  INPUT number
  total ← total + number
NEXT counter
OUTPUT "The average is:" total/50

```

0478/22 October/November 2023

- 3 Four pseudocode statements and five pseudocode uses are shown.

(a) Draw **one** line to link each pseudocode statement to the most appropriate pseudocode use.

Not all pseudocode uses will be required.

Pseudocode statement	Pseudocode use
CALL Colour(NewColour)	counting
Value ← (A1 + A2 + A3) / 3	finding an average
Loop1 ← Loop1 + 1	totalling
IF Count > 7 THEN X1 ← 0	using a conditional statement
	using a procedure

0478/22 May/June 2024

[4]

- 6 Totalling and counting are standard methods of solution.

Numbers are input. The number 9999.9 is the last number to be input and is ignored.

- (a) Write an algorithm in pseudocode to total the numbers input and to output the total. You do **not** need to validate the input.

0478/22 May/June 2023

- 7 An algorithm has been written in pseudocode to calculate a check digit for a four-digit number. The algorithm then outputs the five-digit number including the check digit. The algorithm stops when -1 is input as the fourth digit.

```

01 Flag ← FALSE
02 REPEAT
03   Total ← 0
04   FOR Counter ← 1 TO 4
05     OUTPUT "Enter a digit ", Counter
06     INPUT Number[Counter]
07     Total ← Total + Number * Counter
08     IF Number[Counter] = -1

```

8.1.5 String Handling

- The first character in a string can be in position 0 or 1 but it is usually considered to be in position 1.

	Format	Explanation
Length	<code>LENGTH ("<string>")</code> OR <code>LENGTH (<identifier>)</code>	It outputs the number of characters in the string. (spaces are also included in the length)
Substring	<code>SUBSTRING ("<string>", <start>, <length>)</code> OR <code>SUBSTRING (<identifier>, <start>, <length>)</code>	The first number is the first character of the text you want to isolate and the second number is the final character you want to include.

Upper	UCASE (" <string> ") OR UCASE (<identifier>)	It outputs all the characters in uppercase irrespective of if they were originally in upper or lower case.
Lower	LCASE (" <string> ") OR LCASE (<identifier>)	It outputs all the characters in lowercase irrespective of if they were originally in upper or lower case.

- 7 (a) The string operation `SUBSTRING(FullText, X, Y)` returns a string from `FullText` beginning at position `X` that is `Y` characters long. The first character in `FullText` is in position 1.

Write the pseudocode statements to:

- store the string "IGCSE Computer Science at Cambridge" in `FullText`
- extract and display the words "Computer Science" from the string and store it in a suitable variable
- output the original string in upper case.

```
FullText ← "IGCSE Computer Science at Cambridge" PartText←
SUBSTRING(FullText, 7, 16)
OUTPUT PartText
```

OUTPUT UCASE(FullText)

0478/22 October/November 2023

- 7 The string operation `SUBSTRING(Quote, Start, Number)` returns a string from `Quote` beginning at position `Start` that is `Number` characters long. The first character in `Quote` is in position 1.

Write pseudocode statements to:

- store the string "Learning Never Exhausts The Mind" in `Quote`
- extract and display the words "The Mind" from the string
- output the original string in lower case.

0478/21 May/June 2023

- 8 The function `LENGTH(Phrase)` calculates the length of a string `Phrase`

(a) Write the pseudocode statements to:

- store the string "The beginning is the most important part" in `Phrase`
- calculate and output the length of the string
- output the string in upper case.

(b) Write the output your pseudocode should produce.

0478/22 May/June 2024

- (b) The program design will include a pseudocode algorithm. Assume that the product code is stored in the variable `Product`

(i) Write the pseudocode to make sure that the product code is six characters in length.

8.1 Programming concepts continued

Candidates should be able to:

- (f) Understand and use arithmetic, relational and logical operators

Notes and guidance

- Arithmetic, limited to:
 - +
 - −
 - /
 - *
 - ^ (raised to power of)
 - MOD
 - DIV

- Relational, limited to:

- =
- <
- <=
- >
- >=
- <> (not equal to)

- Logical, limited to:

- AND
- OR
- NOT

5 Understand and use nested statements

- Including nested selection and iteration
- Candidates will **not** be required to write more than three levels of nested statements

6 (a) Understand what is meant by procedures, functions and parameters

(b) Define and use procedures and functions, with or without parameters

(c) Understand and use local and global variables

- Procedures and functions may have up to three parameters

7 Understand and use library routines

- Including:
 - MOD
 - DIV
 - ROUND
 - RANDOM

8.1.6 Arithmetic Operators

Operator	Explanation	Example
+	Addition	2+3 is 5
-	Subtraction	2-3 is -1

/	Division (The resulting value should be of data type REAL, even if the operands are integers.)	4/2 is 2
*	Multiplication	6*7 is 42
^	Raised to the power of	2^4 is 16
MOD	The remainder that is left over when one number is divided by another. (Remainder)	5MOD2 is 1
DIV	Used to find the quotient (integer number before the decimal point) after division. (Quotient)	5DIV2 is 2
ROUND	• Rounds the number to the specified decimal places • The result is rounded to the next highest or the next lowest value	ROUND (3.2154, 3) gives 3.215
RANDOM	Returns a number between 0 and 1 inclusive.	RANDOM ()

8.1.7 Relational Operators

The result of these operations is always of data type **BOOLEAN**.

Operators	Explanation
> < >=	Greater than
<= = <>	Less than
	Greater than or equal to
	Less than or equal to
	Equal to
	Not equal to

8.1.8 Relational / Logical Operators

The operands and results of these operations must always be of data type **BOOLEAN**.

Operator	Explanation
AND	Both parameters must be true

NOT	The parameter must be false
OR	Either or both parameters must be true

8.1.9 i) Procedures and Functions

Local Variables:

- Their scope is a defined block of code/subroutine/procedure/function.
- Their value cannot be changed elsewhere in the program.

Global Variables:

- Their scope is the whole program.
- Their value can be changed anywhere in the program.

Parameters:

Values that are sent to a subroutine.

Procedures:

- It is a subroutine that does not return a value.
- It can have up to 3 parameters in this syllabus.
- The variables used in it are local variables.

Procedure without parameters

```
PROCEDURE <identifier>()
  <statement(s)>
ENDPROCEDURE
```

```
CALL <identifier>()
```

Procedure with parameters

```
PROCEDURE <identifier>(<param1> : <data type>, <param2> : <data
type>...)
  <statement(s)>
ENDPROCEDURE
```

```
CALL (value1, value2, ...)
```

Q. Define and call a procedure that calculates the tip on an inputted value. (the tip is set at 20%). Use an input of 70.

```
PROCEDURE Tip (price : REAL)
  DECLARE tipamount
  tipamount ← value * 0.2
  OUTPUT "The tip amount is ", tipamount
ENDPROCEDURE
```

```
CALL Tip(70)
```

Q. (a) Define a procedure named Average that calculates the average of 2 numbers.

(b) Write the pseudocode to use a procedure named Total that passes the values 25 and 50 to the procedure.

Functions:

- It is a subroutine that always returns a value.
- It can have up to 3 parameters according to the syllabus.
- The variables used within it are local variables.
- It is called with an expression.

Function without parameters

```
FUNCTION <identifier>() RETURNS <data type>
  <statement(s)>
ENDFUNCTION
```

```
<identifier> //calling the procedure
```

Function with parameters

```
FUNCTION <identifier>( <param1> : <data type>,  
<param2> : <data type>...) RETURNS <data type>  
    <statement(s)>  
ENDFUNCTION
```

```
<identifier> (param1, param2...)//calling the procedure
```

0478/22 February/March 2023

Q. 10 The variables X, Y and Z are used in a program: X stores a whole number, Y stores a decimal number and Z stores a flag that can be set to TRUE or FALSE.

(b) The function Same(A,B) returns TRUE if the value of A is the same as the value of B when B is rounded to the nearest whole number and FALSE otherwise. Write pseudocode statements to:

- define the function
- call the function with X and Y and store the return value in Z

```
FUNCTION Same(A : INTEGER, B : REAL) RETURNS BOOLEAN  
    IF A = ROUND(B,0) THEN  
        RETURN TRUE  
    ELSE  
        RETURN FALSE  
    ENDIF ENDFUNCTION
```

```
Z ← Same(X,Y)
```

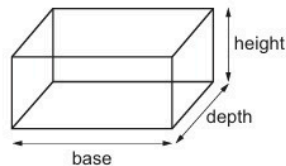
Description	Programming concept
a subroutine that may not return a value	function
a value that is declared and used within a specific procedure	procedure
a value that a procedure expects you to supply when it is called	parameter
a subroutine that will always return a value	global variable
	local variable

0478/22 May/June 2025

- 10 A program is required to allow students to calculate the volume of a number of standard three-dimensional shapes. It will use functions to allow other shapes to be added in the future.

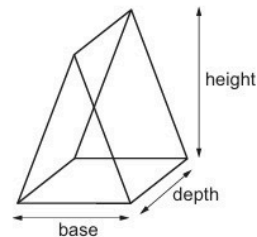
Cuboid

$$\text{volume} = \text{base} \times \text{depth} \times \text{height}$$



Triangular prism

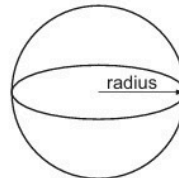
$$\text{volume} = \frac{1}{2} \times \text{base} \times \text{depth} \times \text{height}$$



Sphere

$$\text{volume} = \frac{4}{3} \times \text{Pi} \times \text{radius} \times \text{radius} \times \text{radius}$$

Pi is a constant with value 3.142



- Create one function for each shape to calculate and return the volume using the given formulae. Each function uses the input data as parameters.

8.1.9 ii) Library Routines

Library Routine	Explanation	Example
MOD	Returns the remainder of the division.	DIV(21, 4) gives 1
DIV	• Performs integer division. • Only the whole number part of the answer is retained. (quotient)	DIV(21, 4) gives 5
ROUND	• Rounds the number to the specified decimal places • The result is rounded to the next highest or the next lowest value	ROUND(3.2154, 3) gives 3.215
RANDOM	Returns a number between 0 and 1 inclusive.	RANDOM()

Q. Write a pseudocode statement to generate a random number between 1 and 6 and store it in DiceRoll.

```
DiceRoll ← ROUND(RANDOM() * 6, 0) + 1
```

Q. Write a program to output a single random digit from 1 to 10 inclusive.

8.2 Arrays

Candidates should be able to:

- 1 Declare and use one-dimensional (1D) and two-dimensional (2D) arrays
- 2 Understand the use of arrays
- 3 Write values into and read values from an array using iteration

Notes and guidance

- Including the use of variables as indexes in arrays
- The first index can be zero or one
- Including nested iteration

8.3 File handling

Candidates should be able to:

- 1 Understand the purpose of storing data in a file to be used by a program
- 2 Open, close and use a file for reading and writing

Notes and guidance

- Including:
 - read and write single items of data
 - read and write a line of text

8.2 Arrays

8.2.1 Declaration

Syntax:

```
1 DECLARE <identifier> : ARRAY[<l>:<u>] OF <data type>
```

```
1 DECLARE StudentName : ARRAY[1:30] OF STRING
2 StudentName[19] ← "John Doe"
```

Main points

- 1D arrays are used to store a list of data values of the same type.
- Each element is accessed using one index
- <l> is the lower bound, <u> is the upper bound.
- Arrays make it easier to handle many variables of the same type.

8.2.2 Use of Arrays

Concept of 1D and 2D arrays

Let's say the question wants you to input the marks of 2 students

```
1 DECLARE MarkA, MarkB : INTEGER
2 OUTPUT "Enter marks of student A"
3 INPUT MarkA
4 OUTPUT "Enter marks of student B"
5 INPUT MarkB
```

Now 20 students – that's a lot harder to handle isn't it? Instead of making 20 different variables, let's use a **1D array** instead!

```
1 DECLARE StudentMarks : ARRAY[1:20] OF INTEGER
2 FOR StudentNo ← 1 TO 20
3   OUTPUT "Enter marks of student no. ", StudentNo
4   INPUT StudentMarks[StudentNo]
5 NEXT StudentNo
```

Now what if each student takes more than one subject? Well instead of making a 1D array for each subject, we can use **2D arrays**.

```
1 DECLARE StudentMarks : ARRAY[1:20, 1:10] OF INTEGER
2 FOR StudentNo ← 1 TO 20
3   FOR SubjectNo ← 1 TO 10
4     OUTPUT "Enter marks of student no. ", StudentNo
5     OUTPUT "For subject no. ", SubjectNo
6     INPUT StudentMarks[StudentNo, SubjectNo]
7   NEXT SubjectNo
8 NEXT StudentNo
```

Main points

- Arrays are used to store a collection of values, instead of making too many variables
- 1D arrays store values one by one, for a limited length
- 2D arrays are more like a table/grid, having rows and columns of where values can be inserted into

8.2.3 Writing and Reading

Writing values

- Using **INPUT**
 - **INPUT** ArrayName[Index] for 1D
 - **INPUT** ArrayName[Row, Column] for 2D
- Using assignment
 - ArrayName[Index] ← Expression for 1D
 - ArrayName[Row, Column] ← Expression for 2D

Reading values

You can reference the array by

- ArrayName[Index] for 1D
- ArrayName[Row, Column] for 2D

Treat it like a normal variable.

8.3 File Handling

8.3.1 Purpose of storing data in a file

Exam question:

```
1 OPENFILE <filename> FOR <mode>
```

Modes:

- **READ** → Read data from file
- **WRITE** → Write data to file (overwrites existing data)

Examples:

```
1 OPENFILE "Names.txt" FOR READ  
  OPENFILE "Remarks.txt" FOR WRITE
```

Reading and writing:

```
1 READFILE "Names.txt", Data  
  WRITEFILE "Remarks.txt", Remark
```

Closing a file:

```
1 CLOSEFILE "Names.txt"
```

Main points:

- Always open a file before reading or writing.
- Use the correct mode (**READ** or **WRITE**).
- Always close the file after finishing.

Give **two** reasons for storing data in a file.

Answer:

One mark for each point (max two)

- Data is stored permanently
- Data can be moved to another computer
- Another copy of data can be made and stored//accessed elsewhere // backup copy

To put it simply, you would want to store data in files because you want to keep the data when you're done using the program and allow other software to access the files (so then you can move it to another computer, make a backup, etc.).

8.3.2 File Commands

Syntax:

Opening a file

File handling Past Paper (P23/M/J/23):

Write the pseudocode statements to:

- allow a string to be input to the variable `Saying`
- store the content of the variable `Saying` in a text file named `"Quotations.txt"`
- make sure the text file is closed at the end of the algorithm.

Answer:

One mark per mark point, max **five**

- MP1 input a string into `Saying`
- MP2 correct use of `OPENFILE` to write data
- MP3 correct use of `WRITEFILE` to write `Saying`
- MP4 correct use of `CLOSEFILE`
- MP5 correct use of filename `"Quotations.txt"` throughout

Example:

```
1  OUTPUT "Enter the saying"
2  INPUT Saying
3  OPENFILE "Quotations.txt" FOR WRITE
4  WRITEFILE "Quotations.txt", Saying
5  CLOSEFILE "Quotations.txt"
```

9 Databases

9 Databases

Candidates should be able to:

- 1 Define a single-table database from given data storage requirements
- 2 Suggest suitable basic data types
- 3 Understand the purpose of a primary key and identify a suitable primary key for a given database table
- 4 Read, understand and complete structured query language (SQL) scripts to query data stored in a single database table

Notes and guidance

- Including:
 - fields
 - records
 - validation
- Including:
 - text/alphanumeric
 - character
 - Boolean
 - integer
 - real
 - date/time
- Limited to:
 - SELECT
 - FROM
 - WHERE
 - ORDER BY DESCENDING
 - ORDER BY ASCENDING
 - SUM
 - COUNT
 - AND
 - OR
- Identifying the output given by an SQL statement that will query the given contents of a database table

- A database is is an organized collection of data which enables efficient storage, retrieval, and management of information.
- A single-table database has the following components: fields (no. of columns) and records (no. of rows).

- To differentiate between data items, there is a concept known as the primary key; a field that uniquely identifies each record in a table.
- To ensure data integrity, validation rules that are used as checks.
- Each field can hold specific kinds of data, falling under the category of data types.

Data Type	Explanation	Example
Integer/Number	Letters, numbers, or symbols.	67,526
REAL	Used for money, measurements, etc.	42.6,-8.0
Character	Single letter, number, or symbol	'x','#'
Text/String/Alphanumeric BOOLEAN	Letters, numbers, or symbols	"Hello world"
	Two possible values (e.g. Yes/No)	YES ; NO
Date/Time	Dates and/or times in a standardized format	09/12/2005

9.1 SQL (Structured Query Language)

- It is used to create, update, query, and manage databases.
- Allowing you to extract and update data from database tables.

- Format:

- ❖ **SELECT** field1, field2, ...
 - ❖ **FROM** table_name
 - ❖ **WHERE** condition
 - ❖ **ORDER BY** field ASC/DESC;

- Note: asc/desc refers to ascending/descending order.
 - Note: A semi-colon is necessary to add at the end.
 - Note: An order statement is not necessary unless stated. (same case as with WHERE)

- Example:

- ❖ SELECT Name, Age
 - ❖ FROM Class
 - ❖ WHERE Age>14
 - ❖ ORDER BY Age DESC;

- You can apply two or more conditions in a WHERE clause using:

AND (both conditions must be true)
OR (at least one condition must be true)

Examples:

WHERE Class = '11B' AND Age >= 16;

WHERE Price < 600 OR Manufacturer = 'Apple';

Combination of AND, OR:

WHERE (Job = 'Manager' AND City = 'Lahore')
OR Salary > 500000;

- **Use of SUM and COUNT**

- SUM clause adds values, while the COUNT clause numerates items.

- Examples:

SELECT COUNT(Students)
FROM Council
WHERE Marks > 85;

SELECT SUM(Salary)
FROM Employees;

Past Paper Questions:

Q1.

A database table, TRAIN, is to be set up for a railway company to keep a record of the engines available for use. Each engine has a unique number made up of 5 digits, nnnnn. The engines are classified as freight (F) or passenger (P) together with a power classification that is a whole number between 0 and 9, for example F8. The railway company keeps a record of the date of the last service for each engine.

- a) Identify the three fields required for the database. Give each field a suitable name and data type. Provide a sample of data that you could expect to see in the field.

Field 1 Name

Data type

Data sample

Field 2 Name

Data type

Data sample

Field 3 Name

Data type

Data sample[6]

- b)

State the field that you should choose as the primary key.

.....[1]

- c) Using the query-by-example grid below, write a query to identify all passenger engines that have not been serviced in the past 12 months. Only display the engine numbers.

Field:				
Table:				
Sort:				
Show:	☐	☐	☐	☐
Criteria:				
or:				

[3]

Q2.

A convenience store which sells general groceries wants to set up a database table called STOCK. The table will contain fields including a description of the item, the price of the item and the number in stock for each item. The STOCK table also has a fourth field to be used as a primary key.

- a) Complete the table to suggest a suitable field name for each of the four fields in the table STOCK. Give the purpose of the data to be stored in each field.

Field name	Purpose of field contents

[4]

b) Complete the query-by-example grid to output stock items where the quantity in stock has fallen below 20. Only show the primary key and description of the items.

Field:				
Table:				
Sort:				
Show:	☐	☐	☐	☐
Criteria:				
or:				

[3]

Q3.

A sanctuary for pheasants has set up a new database table called `PheasantList` to store details of the different species of bird at the sanctuary. Part of this table is given, showing: species, description, number of birds at the sanctuary, if the birds are breeding or **not**, and number of young born this year.

Species	Description	NumberBirds	Breeding	Young
Edwards	blue-black with white tail	5	Yes	0
Japanese green	dark green with pale grey tail	2	Yes	2
Reeves	golden, white and red scaled plumage	4	Yes	1
Crawfords Kalij	glossy blue-black plumage	4	No	0
Crested fireback	blue-black with black tail	3	No	0
True silver	white laced top half and black lower half	7	Yes	1
Siamese fireback	grey plumage with crimson legs and feet	5	No	0
Mikado	iridescent plumage with white striped wings	3	Yes	4
Red junglefowl	many colours	2	Yes	0
Himalayan monal	many colours with metallic green crest	3	Yes	2
White eared	white with ear tufts	5	Yes	3
Brown eared	brown with ear tufts	9	Yes	1
Ring necked	long tail with white ring neck	2	Yes	2
Golden	rainbow coloured	3	Yes	4

(a) State the number of records and fields in this part of the database table.

Records

Fields

[2]

(b) (i) Give the name of a field that could be used for the primary key.

..... [1]

(ii) Explain why the sanctuary might decide **not** to use the field in (b)(i) as the primary key.

.....

..... [1]

(iii) A new field `SpeciesID` is added to the database table.
This field contains a six-character code, for example Ph0001.

Give a reason why this field would be a better primary key.

.....

..... [1]

(c) Write the output that would be given by this structured query language (SQL) statement:

```
SELECT Species, Description
FROM PheasantList
WHERE NumberBirds > 6;
```

.....
..... [2]

(d) Complete this SQL statement to display all the species of pheasant where the birds are breeding and there were no young born this year:

```
SELECT .....
FROM .....
WHERE .....
..... ;
[4]
```

10 Boolean Logic

Candidates should be able to:

- 1 Identify and use the standard symbols for logic gates
- 2 Define and understand the functions of logic gates

Notes and guidance

- See section 4 for logic gate symbols
- Including:
 - NOT
 - AND
 - OR
 - NAND
 - NOR
 - XOR (EOR)
 - the binary output produced from all the possible binary inputs
- NOT is a single input gate
- All other gates are limited to two inputs

Logic Gates:

They are the fundamental building blocks of all digital circuits, performing Boolean logic operations on one or more binary inputs to produce a single binary output.

NOT

The only single input gate

The only single-input gate				
Expression	Symbol	Function	Truth Table	
NOT A		Returns the opposite of the inputted value.		
			Input	Output
			1	0
			0	1

AND

Expression	Symbol	Function	Truth Table															
A AND B		It returns True if both the inputs are true or 1.	<table><tr><th>Input A</th><th>Input B</th><th>Output</th></tr><tr><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td></tr></table>	Input A	Input B	Output	1	1	1	1	0	0	0	1	0	0	0	0
Input A	Input B	Output																
1	1	1																
1	0	0																
0	1	0																
0	0	0																

OR

Expression	Symbol	Function	Truth Table															
A OR B		It returns True if either of the inputs are false or 0.	<table><tr><th>Input A</th><th>Input B</th><th>Output</th></tr><tr><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>0</td><td>0</td><td>0</td></tr></table>	Input A	Input B	Output	1	1	1	1	0	1	0	1	1	0	0	0
Input A	Input B	Output																
1	1	1																
1	0	1																
0	1	1																
0	0	0																

NAND

Expression	Symbol	Function	Truth Table															
A NAND B		It returns True if either or both of the inputs are false or 0.	<table><tr><th>Input A</th><th>InputB</th><th>Output</th></tr><tr><td>1</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>0</td><td>0</td><td>1</td></tr></table>	Input A	InputB	Output	1	1	0	1	0	1	0	1	1	0	0	1
Input A	InputB	Output																
1	1	0																
1	0	1																
0	1	1																
0	0	1																

NOR

Expression	Symbol	Function	Truth Table															
A NOR B		It returns True if both of the inputs are false or 0.	<table><tr><th>Input A</th><th>Input B</th><th>Output</th></tr><tr><td>1</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>0</td><td>0</td><td>1</td></tr></table>	Input A	Input B	Output	1	1	0	1	0	0	0	1	0	0	0	1
Input A	Input B	Output																
1	1	0																
1	0	0																
0	1	0																
0	0	1																

XOR(EOR)

Expression	Symbol	Function	Truth Table															
A XOR B		It returns True if either of the inputs are true(1) or false(0); but not if both are true or false.	<table><tr><th>Input A</th><th>InputB</th><th>Output</th></tr><tr><td>1</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>0</td><td>0</td><td>0</td></tr></table>	Input A	InputB	Output	1	1	0	1	0	1	0	1	1	0	0	0
Input A	InputB	Output																
1	1	0																
1	0	1																
0	1	1																
0	0	0																

0478/23 May/June 2024

Combining Logic Gates:

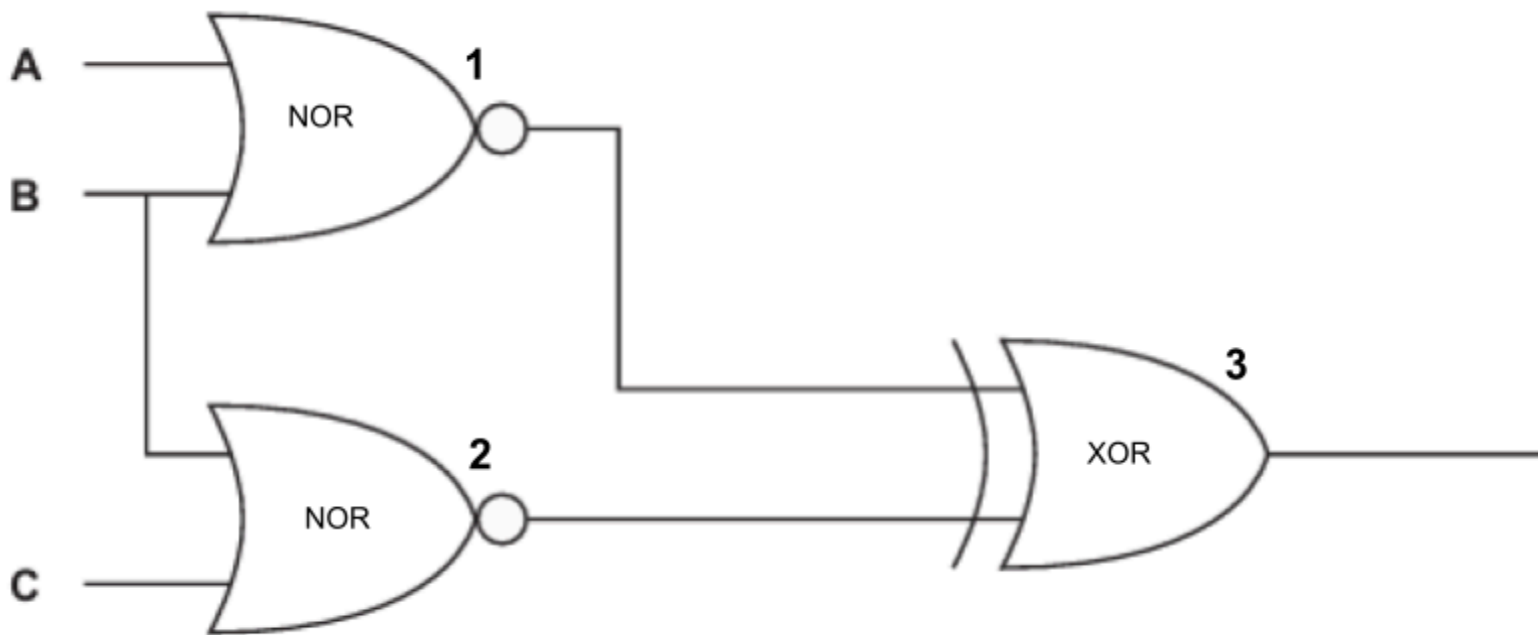
2 (a) Four logic gate symbols and five logic functions are shown.

Draw **one** line to link each logic gate symbol to the appropriate logic function.

Not all logic functions will be used.

Logic gate symbol

Logic function



We can combine these logic gates step by step:

1. $A \text{ NOR } B$ 2. $B \text{ NOR } C$
3. $A \text{ NOR } B \text{ XOR } B \text{ NOR } C$

So the output would be: $(A \text{ NOR } B) \text{ XOR } (B \text{ NOR } C)$

- You are not required to simplify these expressions.

0478/22 February/March 2024

7 Consider the logic expression:

$$X = (A \text{ XOR } B) \text{ AND } (\text{NOT } B \text{ AND } C)$$

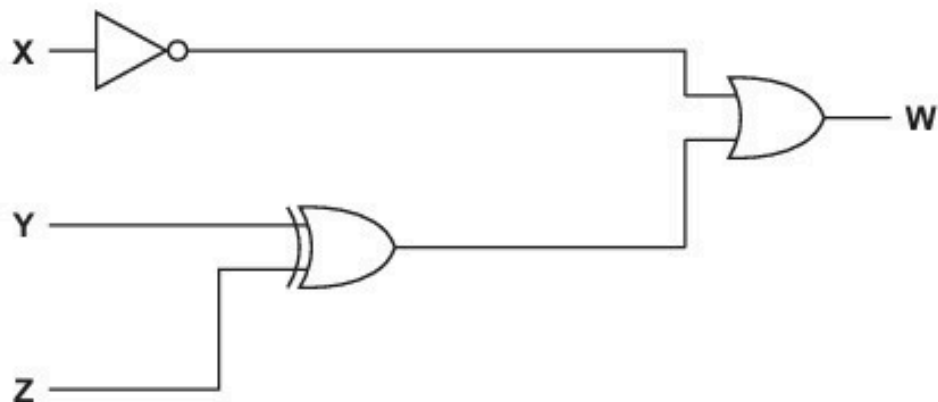
- (a) Draw a logic circuit for this logic expression. Each logic gate must have a maximum of **two** inputs. Do **not** simplify this logic expression.



[4]

0478/21 October/November 2024

7 Consider the logic circuit:



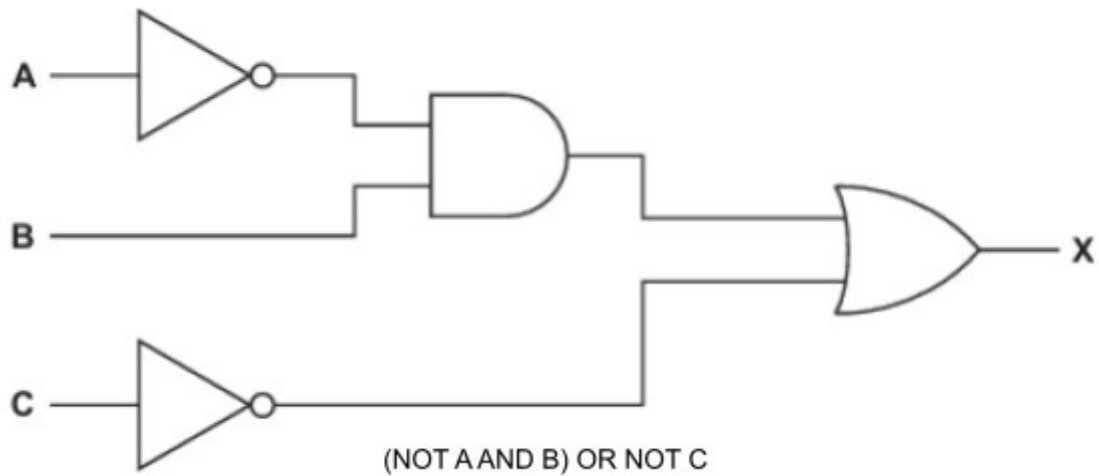
- (a) Write a logic expression for the given logic circuit. Do **not** attempt to simplify the logic expression.

W =

[3]

Truth Tables

- A truth table is a diagram of the outputs from all possible combinations of input.



A	B	C	NOT A	NOT C	NOT A AND B	X
0	0	0	1	1	0	1
0	0	1	1	0	0	0
0	1	0	1	1	1	1
0	1	1	1	0	1	1
1	0	0	0	1	0	1
1	0	1	0	0	0	0
1	1	0	0	1	0	1
1	1	1	0	0	0	0

9 Consider this logic expression:

$$Z = (A \text{ NAND } B) \text{ XOR } (\text{NOT } (\text{NOT } B \text{ NAND } C))$$

(a) Draw a logic circuit for this logic expression.

Each logic gate must have a maximum of **two** inputs.

Do **not** simplify the logic expression.



[5]

(b) Complete the truth table from the given logic expression.

A	B	C	Working space	Z
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

7 Consider the logic expression:

$$X = (A \text{ AND } B) \text{ XOR } (B \text{ OR NOT } C)$$

(a) Draw a logic circuit for this logic expression.

Each logic gate must have a maximum of **two** inputs.

Do **not** simplify the logic expression.



[4]

(b) Complete the truth table for the given logic expression.

A	B	C	Working space	X
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

[4]

Forming Logic Expressions

(b) Consider the rules given in the following table:

Input	Binary value	Description
L	0	dark
	1	light
M	0	no motion detected
	1	motion detected
S	0	no sound detected
	1	sound detected

A security system uses sensors to sound an alarm if an intruder is detected.

The alarm sounding is represented by the output **A = 1**.

The alarm sounds if it is dark and if either motion or sound or both motion and sound are detected.

Recognise the keywords in this question:

- If either: OR
- And: AND

The expression would be:

NOT L AND (M OR S OR (M AND S))

0478/23 October/November 2024

- 8 A logic circuit is to be built to control the automatic opening of a window. The window **W** opens if the temperature **T** is too high, the heater **H** is off, and the air conditioning **A** is off.

air conditioning A	air conditioning off	0
	air conditioning on	1
temperature T	not too high	0
	too high	1
heater H	heater off	0
	heater on	1
window W	window closed	0
	window open	1

Complete the truth table for this problem.

A	T	H	W
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

[4]

