

**COMPUTER  
SCIENCE  
REVISION  
NOTES**

**FOR OCR GCSE (9-1)  
SIMPLE, CLEAR & MEMORABLE**

**Ronaldo Butrus**



# CONTENTS

1.1	System Architecture .....	5
1.2	Memory .....	7
1.3	Storage.....	8
1.4	Wired and Wireless Networks.....	9
1.5	Network Topologies, Protocols and Layers.....	11
1.6	System Security .....	14
1.7	Systems Software .....	15
1.8	Ethical, Legal, Cultural and Environmental Concerns.....	16
2.1	Algorithms .....	18
2.2	Programming Techniques .....	21
2.3	Producing Robust Programs .....	24
2.4	Computational Logic .....	26
2.5	Translators and Facilities of Languages.....	27
2.6	Data Representation .....	28

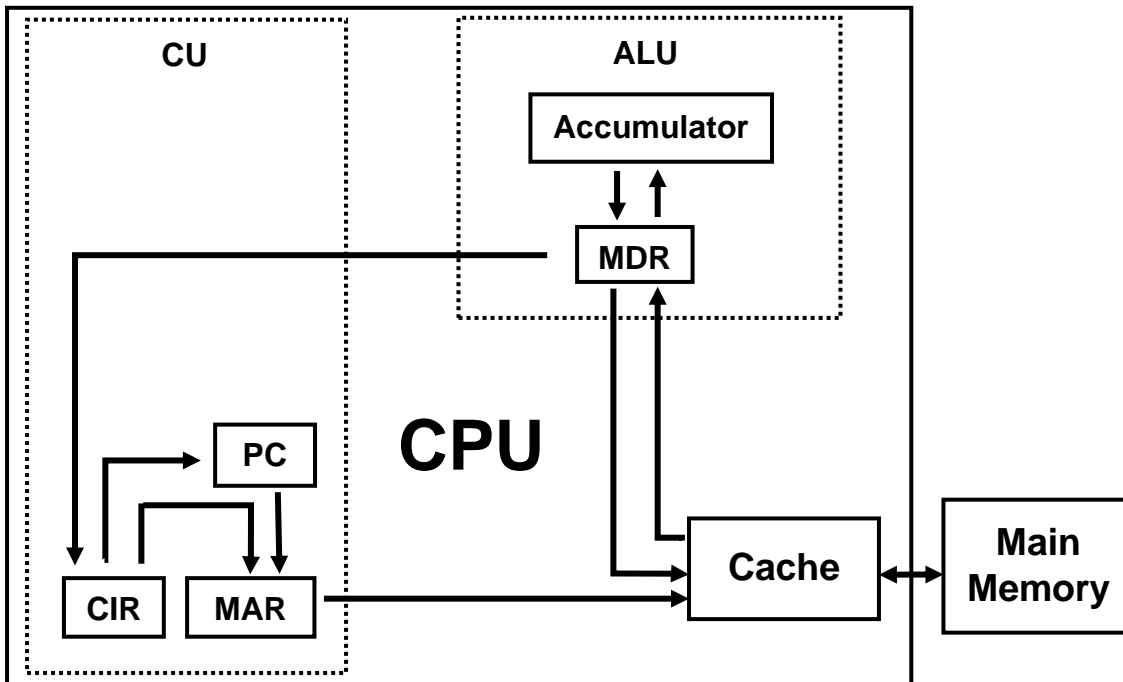
**Using this book**

# USING THIS BOOK

## 1.1 This is a specification chapter

## 1.1 System Architecture

- The purpose of the CPU:
  - executes programs
  - manages hardware
- Von Neumann Architecture:



- **MAR (Memory Address Register):** holds address of instruction/data to be fetched
- **MDR (Memory Data Register):** temporarily holds instruction/data just fetched from main memory
- **CIR (Current Instruction Register):** holds current instruction being executed
- **PC (Program Counter):** holds address of next instruction to be processed
- **ACC (Accumulator):** temporarily holds data and results of operations
- **ALU (Arithmetic Logic Unit):** carries out logical, shift and arithmetic operations on data
- **CU (Control Unit):** coordinates activities in the CPU, by decoding and executing instructions, regulating processor timing, and sends/receives control signals from other parts of the computer
- **Cache:** special high-speed memory that stores recently used and frequently used instructions/data
- Types of cache:
  - **Level 1:** extremely fast, small, holds instructions
  - **Level 2:** fairly fast and medium sized, holds data
  - **Level 3:** fast and larger, improves performance of L1 and L2 cache

## 1.1 - Systems Architecture

- The fetch-execute cycle:
  1. Instruction is fetched:
    - address copied from PC to MAR
    - PC incremented and instruction copied from main memory (address in MAR) to MDR
    - instruction copied from MDR to CIR
  2. Instruction is decoded by the CU
  3. Instruction is executed
  4. The cycle repeats using the address in PC
  
- Factors affecting CPU performance:
  - **clock speed:** number of fetch-execute cycles per second, in Hertz (Hz)
  - **cache size:** more cache means faster access to frequently used instructions and less access to main memory
  - **number of cores:** more cores allow simultaneous execution of instructions
  
- Embedded systems:
  - a computer system that forms part of an electronic device
  - used to perform simple, repeated tasks
  - used in: car stereos, aircrafts, ovens, microwaves, fridges, televisions

## 1.2 Memory

RAM (Random Access Memory)	ROM (Read Only Memory)
<ul style="list-style-type: none"> <li>- <b>volatile</b> (data is lost when power is turned off)</li> <li>- can be <b>written to or read from</b></li> <li>- <b>stores running programs, data, and parts of the operating system</b></li> </ul>	<ul style="list-style-type: none"> <li>- <b>non-volatile</b> (data is <b>not</b> lost when power is turned off)</li> <li>- can <b>only be read from</b> (written to once)</li> <li>- <b>stores BIOS and bootstrap loader</b> (for starting up)</li> </ul>

- **Virtual Memory** is part of the hard disk that behaves like RAM:
  - data that is being used and programs that are currently running are stored in RAM
  - if there is not enough RAM to store these programs the CU has to swap bits of them between RAM and the hard disk
  - hard disk access speeds are lower than RAM access speeds
  - overall speed decreases and computer performance decreases
  
- **Flash memory** is memory that uses electricity and no moving parts:
  - benefits:
    - more reliable (no moving parts and unaffected by electromagnetic interference)
    - more portable (smaller and lighter)
    - faster access speeds
    - lower power consumption
    - run cooler (no moving parts)
  - drawbacks:
    - costs more

## 1.3 – Storage

### 1.3 Storage

- **Secondary storage** is non-volatile, long term storage, used to store data that is needed to be saved, even after the power is turned off.
- You need to be able to calculate the required capacity of a storage device.

E.g. if you have **250 images**, each with a **file size of 3MB**, which of the following sizes will be the most suitable to use?

**250MB**

**10KB**

**3TB**

**1GB**

**5GB**

$250 \times 3 = 750\text{MB}$

$750\text{MB} = 0.75\text{GB}$

250MB and 10KB are too small; 3TB and 5GB and too big

**1GB is just right.**

**sound file size = sample rate x duration x bit depth**

**image file size = colour depth x image height x image width**

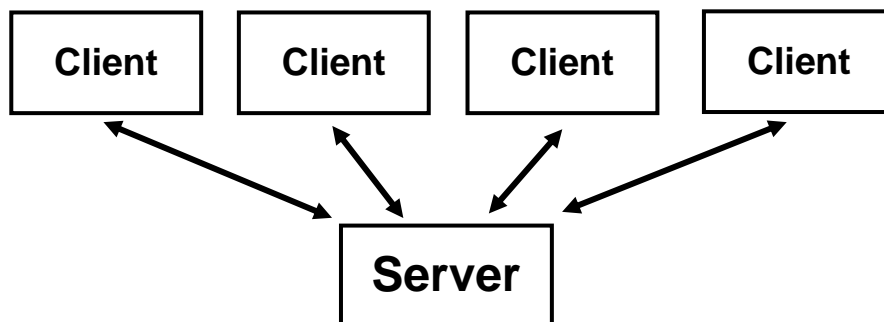
**text file size = bits per character x number of characters**

- Types of secondary storage:
  - **optical:**
    - capacity: *500MB to 25GB*
    - speed: *7MB to 50 MB/s*
    - portability: *easy to carry around*
    - durability: *easily scratched*
    - reliability: *degrades over time*
    - cost: *2p/GB*
  - **magnetic:**
    - capacity: *500MB to 6TB*
    - speed: *500MB to 3GB/s*
    - portability: *fits large pocket*
    - durability: *affected by electromagnetic interference*
    - reliability: *extremely reliable*
    - cost: *0.03p/GB*
  - **solid state:**
    - capacity: *2GB to 2TB*
    - speed: *500MB to 6GB/s*
    - portability: *small and light*
    - durability: *unaffected by magnetism and drops*
    - reliability: *extremely reliable*
    - cost: *25p/GB*

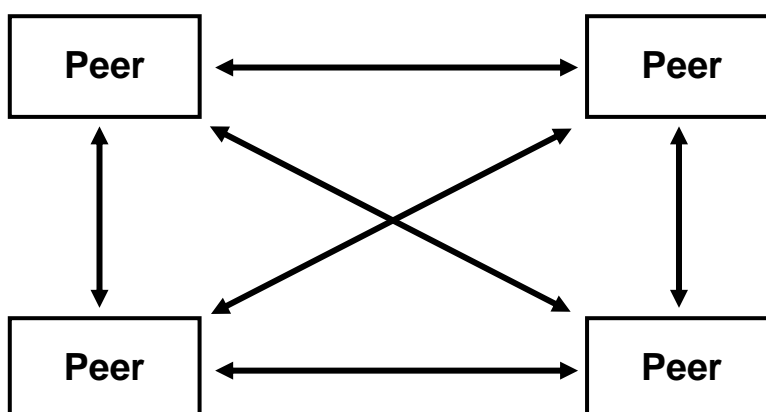


## 1.4 Wired and Wireless Networks

- A **network** is a collection of computers connected together for sharing data, peripherals and an internet connection.
- The **Internet** is a global network of computer networks.
- Types of networks:
  - **Local Area Network (LAN)**: a network of computers on the same site
  - **Wide Area Network (WAN)**: a large network of computers connected by third party resources over a large geographical area
- Factors affecting network performance:
  - **number of devices connected**: more devices mean more traffic
  - **bandwidth**: the amount of data that can travel through a cable at any given time
  - **latency**: the delay between the time an instruction is given to when it is executed
  - **errors in transmission**: errors result in retransmission
- Client-server networks:
  - **server**: powerful computer which provides services or resources, such as storing files, hosting a website, distributing emails and security
  - **client**: a computer used by a user, requesting services from the server



- Peer-to-peer networks:
  - all computers are interconnected with equal access rights
  - file storage, security, backup and email management is done separately

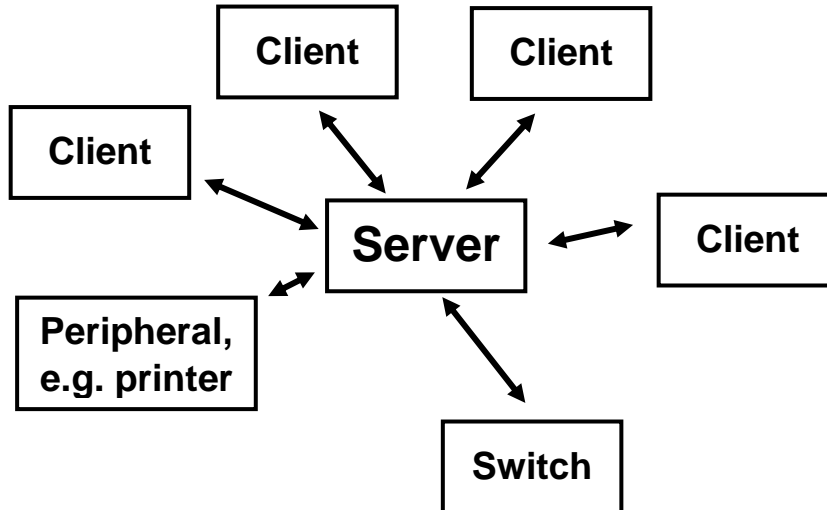


## 1.4 – Wired and Wireless Networks

- Hardware needed to connect computers in a LAN:
  - **Wireless Access Points (WAPs):** a wireless transmitter which receives data from a network via cables and transmits the equivalent radio waves to any device on the network
  - **Router:** acts as a node for transmitting data packets across a WAN
  - **Switch:** forwards inbound packets to individual recipient devices using their unique MAC address
  - **Network Interface Controller/Card (NIC):** enables a device to connect to a network, either wired or wirelessly
  - **Transmission Media:** copper cables (cheapest), fibre optic cables (high bandwidth, strong, unaffected by electromagnetic interference)
- The internet is a worldwide collection of computer networks:
  - **Domain Name Server (DNS):** a database of domain names and IP addresses which can convert domain names/URLs into IP addresses
  - **Hosting:** an internet host stores your web files and makes them available to other Internet-connected devices
  - **The Cloud:** an online storage method so you can access your files globally, and share them via the Internet
- A **virtual network** is a subset of computers connected together as part of a larger physical network.
- E.g. a network between work and home as part a WAN, such as the Internet.

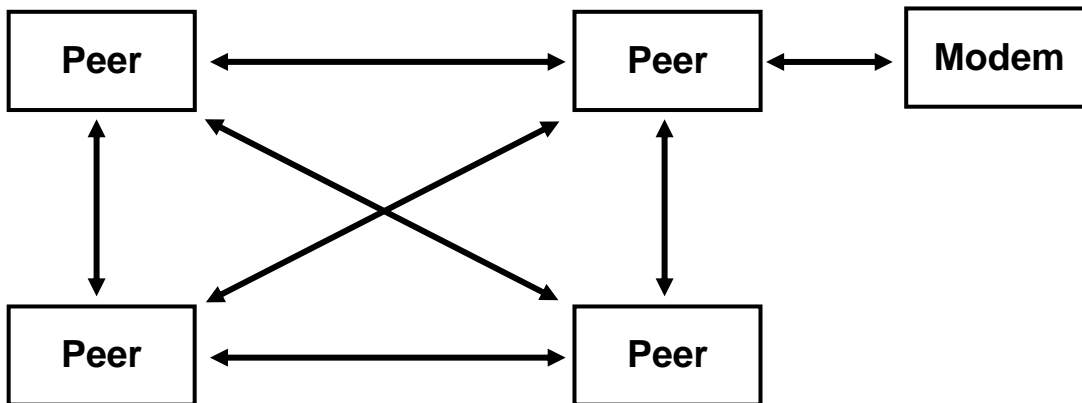
**1.5 Network Topologies, Protocols and Layers**

- Star network:
  - every device is connected to central server
  - server is connected to switch



Advantages of STAR NETWORKS	Disadvantages of STAR NETWORKS
- if one cable fails others are not affected - if one client (computer) fails, others are not affected - consistent performance	- expensive due to extra cables to server - expensive due to extra switch - if server/switch fails, entire network fails

- Mesh networks:
  - every device is connected to every other device
  - each device acts as a node to pass on data



Advantages of MESH NETWORKS	Disadvantages of MESH NETWORKS
- more nodes mean a faster network - new nodes easily added - if one node fails, there are other routes - no need to travel to and from central server	- difficult and slow to construct - hard to maintain/administer - expensive due to excessive cabling

## 1.5 – Network Topologies, Protocols and Layers

- **Wi-Fi (Wireless Fidelity)** features:
  - **frequencies:** 2.4GHz (greater coverage, but less non-overlapping channels) or 5GHz (less crowded, more non-overlapping channels, but not compatible with all devices)
  - **channels:** a 'sub-frequency' with different bandwidths
  - **encryption:** converting plaintext into ciphertext using an encryption algorithm, which you need a special key to reverse
- **Ethernet** is a family of networking protocols which define how devices should format data for transmission between computers on a network.
- An **IP (Internet Protocol) address** is a unique address given to a networked computer, that changes as location changes. It is used to identify the network a computer is connected to and track its whereabouts.
  - **IPv4 (numeric):** uses a 32-bit address scheme allowing for  $2^{32}$  addresses
  - **IPv6 (alphanumeric):** uses a 128-bit address scheme allowing for  $2^{128}$  addresses
- A **MAC (Media Access Control) address** is a unique address given to each NIC that does not change. It is used for forwarding inbound data packets to the intended recipients and for identifying the manufacturer of a product.
- A **protocol** is a principle of a standard to provide rules for areas of computing to allow hardware/software to interact across different manufacturers/producers, including:
  - format of data packets
  - addressing system
  - transmission speed
  - error-checking procedures
- A **communication protocol** is a set of rules for transferring data.
- Commonly used protocols:
  - **TCP/IP (Transmission Control Protocol/Internet Protocol):** defines how messages are broken down and reassembled at their destination and dealing with errors, as well as the route to take
  - **HTTP (Hyper Text Transfer Protocol):** for requesting and receiving access to web pages
  - **HTTPS (HTTP Secure):** encrypts web data to protect from hacks
  - **FTP (File Transfer Protocol):** defines how files are transferred between client and server in networks
  - **POP (Post Office Protocol):** receives and holds emails on a single device
  - **IMAP (Internet Message Access Protocol):** receives and holds emails on a server accessible from multiple devices
  - **SMTP (Simple Mail Transfer Protocol):** sends emails between servers

## 1.5 – Network Topologies, Protocols and Layers

- A **layer** is a division of network functionality.
- Layers are parts of the communication process, each responsible for a specific part.
- For example, when an email is sent over the Internet:
  - **Application Layer:** formatting data for application that receives it
  - **Transport Layer:** divides data into packets and assigns headers
  - **Network Layer:** attaches source and destination IP addresses and calculates routes
  - **Link Layer:** attaches MAC addresses to locate the specific recipient on the network
- Advantages of layers:
  - modular design makes systems easily debuggable
  - suppliers can easily adapt parts of their software for compatibility
- Packet switching in detail:
  1. a file is broken down into data packets of around 512 bytes
  2. each packet is given a header containing:
    - destination IP address
    - source IP address
    - sequence number
    - number of packets in whole communication
    - error checking data
  3. packets are sent along different routes to their destination
  4. errors in transmission result in retransmission
  5. once all packets have reached their destination, they are reordered and reassembled into one file

## 1.6 – System Security

### 1.6 System Security

- Threats posed to devices and systems:

Threat	How attack is used	Purpose
malware	malicious software is entered into a computer system	to disrupt, damage or access a computer system/network
phishing	emails, calls and texts claiming to be a reputable authority	to obtain personal details or payment details
social engineering	using humans as a weak penetration point of a system	to pass malware into a network, obtain data or access a system/network
brute force attacks	trying all possible login details to get into an account	to access an online account or computer system/network
denial of service attacks	preventing access to a website by overloading with useless traffic	to prevent lawful access to a service, system or network
data interception and theft	shouldering, signal interception, looking through rubbish for data	to access a network, system or personal data
SQL injection	inserting malicious SQL code into database fields on websites	to access a database, corrupt websites, or spread viruses
poor network policy	not setting strict guidelines on how a network is used	results in hackers being able to exploit this security weakness

- Vulnerabilities can be identified and prevented by different methods:

Method	What it limits/prevents	How it limits the attack
penetration testing	unauthorised access to device or system	identifies weak points in network for them to be prevented
network forensics	fraudulent use of device or system	identifies improper use of networks, including perpetrators
network policies	poor network policy	setting strict guidelines on how a network may and may not be used
anti-malware software	existence of malware in a system	identifies and destroys malware
firewalls	unauthorised access to/from a network to/from internet sites	restricts access to specific web site addresses and files
user access levels	users exploiting full access to a network by modifying/deleting files	enforces permissions for users to access/modify only relevant files
passwords	unauthorised access to device or system	verifies user by comparing unique login details to those stored in database
encryption	data interception	enciphers transmitted data to make it unintelligible to interceptors

## 1.7 Systems Software

- **System software** is a type of computer program that is designed to run a computer's hardware and application programs.
- Functions of operating systems:
  - **user interface:** allowing a user to interact with computer hardware through the use of windows, icons, menus and pointers (WIMP)
  - **memory management/multitasking:** holds several programs in memory simultaneously and frees up space when a program is terminated
  - **peripheral management and drivers:** managing hardware outside the CPU through drivers, small programs that allow peripherals to interact with the computer
  - **user management:** setting access rights, identifying users on a network, monitoring login and activity, security
  - **file management:** enabling creation, modification, copying, deleting, moving files, searching for files, recording location of files, maintaining access rights to files
- Functions of utility system software:
  - **encryption:** making data meaningless to criminal interceptors by converting plaintext into ciphertext
  - **defragmentation:** moving separate file parts together, allowing faster access and freeing up space
  - **data compression:** reducing the size of files for faster transmission over the Internet by lossy (for pictures) or lossless (for documents) compression
  - **backup:** making a copy of user files which can be restored if corrupted/deleted
    - **full:** copying all user data
    - **incremental:** regularly copying any changed/new data since last backup

## 1.8 – Ethical, Legal, Cultural and Environmental Concerns

### 1.8 Ethical, Legal, Cultural and Environmental Concerns

- Legislation:
  - **Data Protection Act 1998** → personal details must be stored securely; it must:
    - be processed fairly and lawfully
    - be adequate, relevant and not excessive
    - be accurate and up-to-date
    - not be retained for longer than necessary
    - only be used for the purpose it was collected for
    - be kept secure
    - be handled in accordance with people's rights
    - not be transferred outside the EU without protection
  - **Freedom of Information Act 2000** → access to data held by public authorities
  - **Computer Misuse Act 1990** → prevents unauthorised access to programs or data, making the following offences:
    - unauthorised access to computer material
    - unauthorised access with intent to commit/facilitate a crime
    - unauthorised modification of computer material
  - **Copyright Designs and Patent Act 1988** → prevents illegal copying and protects the author, making the following offences:
    - passing a copy to a friend
    - making a copy and selling it
    - using the software on a network (unless the licence allows it)

Open source software	Proprietary software
provides access to source code user has ability to change software	no access to source code commonly purchased as 'off-the-shelf'

- **Freeware** is the same as open source but does not come with the source code.
- **Off-the-shelf software** is generalised.
- **Bespoke software** is customised for a customer.
- A **creative commons licence** gives people the right to:
  - share and use an author's work
  - optionally limit use to non-commercial purposes



## 1.8 – Ethical, Legal, Cultural and Environmental Concerns

Ethical issues:

- contributes to **ill health**
  - contributes to **digital divide**
  - contributes to **social divide**
  - problem of **confidential data** being stored on devices
  - **social pressure on parents** of children who want to upgrade
  - **bullying** of those who cannot afford the latest technology
  - phone manufacturers **intentionally design fragile phones**
  - **high cost** of new devices
- Cultural issues:
- people lose the **ability to interact face-to-face**
  - people can **rate others**
- Environmental issues:
- people **dispose of devices** even if they are in working order
  - equipment sent **abroad to be disposed of**
  - leads to **excessive landfill**
  - **toxic waste** released into land, water and air
  - **waste of resources**
  - **precious metals** are in phones
- Privacy issues:
- **embedded systems** in a variety of products
  - **cookies**
  - **metadata**
  - **hacking**
- Key stakeholders:
- **people worldwide:**
    - **health:** eyesight, mental illnesses from cyberbullying
    - **finance:** spending too much money on technology
    - **social/cultural:** losing ability to interact face-to-face / leaving tradition behind
  - **phone manufacturers:** making huge profits
  - **phone shops/networks:** making huge profits

## 2.1 – Algorithms

### 2.1 Algorithms

- Computational thinking:
  - **abstraction:** removing unnecessary information from a problem to make it easier to solve
  - **decomposition:** breaking down a problem into smaller subparts to make it simpler to solve
  - **algorithmic thinking:** transforming a problem into a series of steps which can be translated into pseudocode or code
- Standard searching algorithms:
  - **binary search:** for sorted lists, this works by repeatedly compares the middle item with the item being searched for and disregarding the half which does not contain the item being searched for until it finds the item
  - **linear search:** for unsorted or sorted lists, this works by comparing every data item from the start to the end until it finds the item

BINARY SEARCH for item with value 8									
3	5	6	8	11	12	14	15	17	18
There are 10 items, the middle item is the 5 <sup>th</sup> or 6 <sup>th</sup> , depending on how you do it (we'll round up). 12 is greater than 8, so we know that we need to look at the lower part of the list.									
3	5	6	8	11					
There are 5 items, the middle item is the 3 <sup>rd</sup> . 6 is less than 8, so we know that we need to look at the upper part of the list.									
8	11								
There are 2 items, the middle item is the 2 <sup>nd</sup> (as we rounded up). 11 is greater than 8, so we know that we need to look at the lower part of the list.									
8									
There is 1 item, the middle item is the 1 <sup>st</sup> . 8 is the item we are looking for – item found!									

LINEAR SEARCH for item with value 8									
3	5	6	8	11	12	14	15	17	18
3 isn't 8, so we go to the next item.									
3	5	6	8	11	12	14	15	17	18
5 isn't 8, so we go to the next item.									
3	5	6	8	11	12	14	15	17	18
6 isn't 8, so we go to the next item.									
3	5	6	8	11	12	14	15	17	18
8 is 8, so item is found!									

- Standard sorting algorithms:
  - **bubble sort:** works by repeatedly comparing adjacent items and swapping them if the first is greater than the second. At the end of each pass, one extra item is in the correct order at the end of the list
  - **merge sort:** works by repeatedly combining sublists (of length 1) to form a sorted list
  - **insertion sort:** works by repeatedly inserting the 'next' item in the sequence into an empty sorted list

BUBBLE SORT								
<b>Original</b>	9	5	4	15	3	8	11	2
<b>Pass 1</b>	5	9	4	15	3	8	11	2
	5	4	9	15	3	8	11	2
	5	4	9	15	3	8	11	2
	5	4	9	3	15	8	11	2
	5	4	9	3	8	15	11	2
	5	4	9	3	8	11	15	2
	5	4	9	3	8	11	2	15
<b>Pass 2</b>	4	5	3	8	9	2	11	15
<b>Pass 3</b>	4	3	5	8	2	9	11	15
<b>Pass 4</b>	3	4	5	2	8	9	11	15
<b>Pass 5</b>	3	4	2	5	8	9	11	15
<b>Pass 6</b>	3	2	4	5	8	9	11	15
<b>Pass 7</b>	2	3	4	5	8	9	11	15

After a maximum of (n-1) passes the list is sorted, where n is the number of items

MERGE SORT								
<b>Original</b>	42, 31, 12, 3, 37, 18, 29, 47							
	Divide the original list into sublists of length 1							
<b>Stage 1</b>	42	31	12	3	37	18	29	47
<b>Stage 2</b>	31, 42		3, 12		18, 37		29, 47	
<b>Stage 3</b>	3, 12, 31, 42				18, 29, 37, 47			
<b>Stage 4</b>	3, 12, 18, 29, 31, 37, 42, 47							

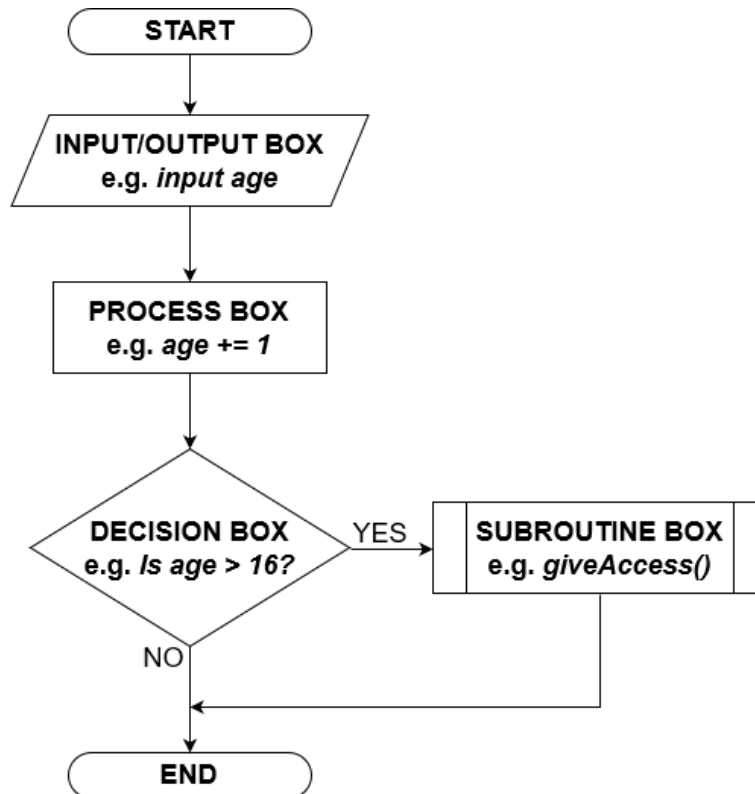
After x in ( $n = 2^x$ ) stages the list is sorted, where n is the number of items

INSERTION SORT								
<b>Original</b>	9	5	4	15	3	8	11	2
<b>Pass 1</b>	9	5	4	15	3	8	11	2
<b>Pass 2</b>	5	9	4	15	3	8	11	2
<b>Pass 3</b>	4	5	9	15	3	8	11	2
<b>Pass 4</b>	4	5	9	15	3	8	11	2
<b>Pass 5</b>	3	4	5	9	15	8	11	2
<b>Pass 6</b>	3	4	5	8	9	15	11	2
<b>Pass 7</b>	3	4	5	8	9	11	15	2
<b>Pass 8</b>	2	3	4	5	8	9	11	15

After a maximum of n passes the list is sorted, where n is the number of items

## 2.1 – Algorithms

- Producing algorithms using:
  - **pseudocode:** refer to section 2.2
  - **flow diagrams:** use the following diagrams



- This specification also tests a candidate's ability to interpret, correct and complete algorithms given appropriate information.
- This program calculates the areas of four triangles and outputs the mean area:

```

total = 0
for count = 1 to 4:
    base = input("Enter base: ")
    height = input("Enter height: ")
    value = (base * height) / 2
    total = total + value
next count
print(total / 4)
  
```

- A trace table records the value of each variable as a program runs, e.g.:

Total	Count	Base	Height	Value
0	1	10	20	100
100	2	8	6	24
124	3	16	12	96
220	4	15	40	300
520				

**2.2 Programming Techniques**

- Key words to remember:
  - **variable:** a memory location used to store a value that can be changed while the program is running
  - **constant:** a memory location used to store a value that does **not** change while the program is running
  - **operator:** an action performable on data, either arithmetic, comparative or string manipulative
  - **input:** data passed into a computer by a user via an input device
  - **output:** data passed to a user by a computer via an output device
  - **assignment:** giving a value to a variable
  
- Programming constructs:

Construct	Definition	Example
Sequence	executing a series of coded steps in a specified order once	<pre>num1 = input "Enter no.1:" num2 = input "Enter no.2:" print num1,"+",num2,"=", (num1+num2)</pre>
Selection	executing a series of coded steps dependant on the validity of a condition(s)	<pre>if userAns == realAns:     print "Well done!" else:     print "Incorrect answer!" endif</pre>
Iteration	executing a series of coded steps an unspecified number of times dependant on the validity of a condition(s)  E.g. while loops, for loops, do-until loops	<pre>goAgain = "Y" while goAgain in ["Y","y"]:     print "Hello, user!"     goAgain = input "Shall I say hello again? (Y/N): " endif  for char in "hello world":     print char.upper()</pre>

- String manipulation:

Pseudocode	Explanation
string.upper	returns uppercase version of string
string.lower	returns lowercase version of string
string.substring(1,3)	returns the 3 characters at (including) index 1 ( <b>slicing</b> )
string[3]	returns character at index 3 (4 <sup>th</sup> character)
string.length	returns number of characters in string (including spaces)
string.index("ab")	returns position of substring "ab" in string
res = string1 + string2	returns addition of string1 and string2 ( <b>concatenation</b> )

## 2.2 – Programming Techniques

- File handling operations:

Pseudocode	Explanation
<code>file = openRead("sample.txt")</code>	open file in read mode assign reference to variable <code>file</code>
<code>x = file.readLine()</code>	stores value of line being read in <code>x</code>
<code>file = openWrite("sample.txt")</code>	open file in write mode assign reference to variable <code>file</code>
<code>file.writeLine("Hello World")</code>	write "Hello World" in <code>sample.txt</code>
<code>while NOT file.endOfFile()</code>	loop continues given that the program has not reached the end of the file
<code>file.close()</code>	closes file

- A **data structure** is a group of data items that can be treated as a set of data.
- A **record** is a 'row' in a database table, storing data of different data types regarding an item (or person).
- This is a data table, called `carTable`:

Registration	Year	Make	Model	Mileage	ULEZCompliant
AB12 CEX	2012	Peugeot	Partner	59783	False
KF68 KJS	2018	Hyundai	Tucson	6752	True
L975 HYH	1993	Mercedes	E200	197291	False
LG08 EHJ	2008	Chevrolet	Matiz	34878	True
DJ19 SJN	2019	Toyota	Aygo	687	True

- Each row is a record, storing a registration, make, model, mileage and ULEZ compliance.
- Each column is a field, storing one attribute as one data type.
- Using SQL:
  - **SELECT** *the fields you want displayed*
  - **FROM** *the table you want to retrieve data from*
  - **WHERE** *the search criteria*
  - you can use **LIKE** *to detect certain characters in a field*
- E.g. to display the mileage of all ULEZ compliant cars made after 2015:
  - **SELECT** Mileage
  - **FROM** `carTable`
  - **WHERE** `ULEZCompliant == True AND year > 2015`
- E.g. to display from all tables the ULEZ compliancy of cars made by Chevrolet with an "E" in their registration:
  - **SELECT** `ULEZCompliant`
  - **FROM** `*`
  - **WHERE** `make == "Chevrolet" AND Make LIKE "%E%"`
- Wildcards:
  - `%` is a substitute for zero or more unspecified characters
  - `*` is a substitute for 'all fields'

## 2.2 – Programming Techniques

- An **array** is a fixed length static structure where:
  - multiple data items
  - of the same data type
  - are stored under one identifier.
- E.g. `names = ["Tom", "Jack", "Rosie", "Sam"]`
- A **2D array** involves an array within a larger array, and can be thought to be like a table.
- E.g. `scores = [["Tom", 99], ["Sarah", 78], ["Jack", 86]]`
- A **function** is a set of reusable code that processes parameters and returns a value.
- A **procedure** is a set of reusable code that performs a specific task.
- Benefits of using subroutines (functions/procedures):
  - reusable (from different programs)
  - easier to find and debug
  - easier to understand / clearer code
- Data types:

Data type	Meaning	Example	Typical amount of memory
Integer	whole number	8	2 or 4 bytes
Real / Float	decimal number	8.724, or even 8.0	4 or 8 bytes
Boolean	True or False	True	1 byte
Character	a single character	"a"	1 byte
String	zero or more characters	"Hello world!"	1 byte per character

- **Casting** means changing the data type of a variable:
  - 1 (integer) can become "1" (string)
  - "10.2" (string) can become 10.2 (float)
  - this can be done using the in-built functions `int()`, `float()` or `str()`
- Common operators:

Arithmetic (give a numerical result)		Comparative (give a Boolean result)	
+	addition	<	less than
-	subtraction	>	greater than
*	multiplication	<=	less than or equal to
/	division	>=	more than or equal to
^ or **	exponentiation (power)	==	equal to
DIV or //	integer division (e.g. 10//3 = 3)	!=	not equal to
MOD or %	modulus (e.g. 10%3 = 1)		

## 2.3 – Producing Robust Programs

### 2.3 Producing Robust Programs

- Defensive design considerations:
  - **input validation:** ensuring input is in the acceptable format, including range, length, type, presence and format
  - **input sanitisation:** 'cleaning up' input by putting it into the format needed
  - **planning for contingencies:** using try...except loops, repeating input etc. to ensure a user's error is corrected
  - **anticipating misuse:** implementing access rights, passwords, masking, encryption etc. to ensure a user cannot misuse a program to access data etc.
  - **authentication:** ensuring the user is legitimate, using a login or optical, facial or fingerprint recognition
  
- Maintainability:
  - **comments:** explain nature and purpose of program and its subroutines
  - **indentation:** shows where selection/iteration statements and subroutines start/end
  
- General purpose of testing:
  - ensures software meets expectations
  - prevents unexpected results
  
- Types of testing:
  - **iterative testing:** testing a section of code as you write it to ensure it works correctly
  - **final/terminal testing:** testing the entire source code to ensure subroutines work correctly once put together
  
- Types of errors:
  - **syntax error:** error which break the grammatical rules of the programming language and stops it from being run/translated
  - **logic error:** errors which produce unexpected output
  
- Types of input for testing:
  - **normal:** data which should be accepted by a program without causing errors
  - **boundary:** data of the correct type which is on the very edge of being valid
  - **invalid:** data of the correct type but outside the accepted validation limit
  - **erroneous:** data of the incorrect type which should be rejected by a program



## 2.3 – Producing Robust Programs

- Selecting and using suitable test data:

```
mark = input("Enter your mark out of 100: ")

if mark >= 87:
    print("GCSE Grade: 9")
elif mark >= 79:
    print("GCSE Grade: 8")
elif mark >= 71:
    print("GCSE Grade: 7")
else:
    print("Don't worry, it's not a great grade!")
```

- we need to test this program for:
  - normal data, e.g. 89, or 65
  - boundary data, e.g. 87 or 79
  - invalid data, e.g. -10
  - erroneous data, e.g. "Eighty"
- Test tables are tables outlining test data alongside the result of each test, e.g.:

No.	Test purpose	Test data	Expected output	Actual output	Solution (if applicable)
1	Normal data	89	GCSE Grade: 9	GCSE Grade: 9	
2	Normal data	65	Don't worry	Don't worry	
3	Boundary data	87	GCSE Grade: 9	GCSE Grade: 9	
4	Boundary data	79	GCSE Grade: 8	GCSE Grade: 8	
5	Invalid data	-1	Error message	Don't worry	Introduce invalid input message
6	Erroneous data	"Lol"	Error message	*Program crashes*	Introduce error message

- revised code:

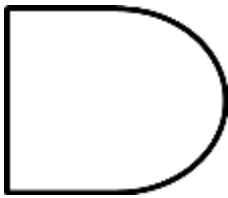

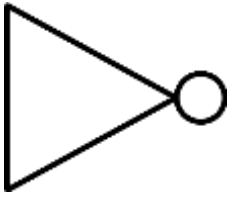
```
mark = input("Enter your mark out of 100: ")

if mark >= 87:
    print("GCSE Grade: 9")
elif mark >= 79:
    print("GCSE Grade: 8")
elif mark >= 71:
    print("GCSE Grade: 7")
elif mark < 71:
    print("Don't worry, it's not a great grade!")
else:
    print("Invalid input, please try again!")
```

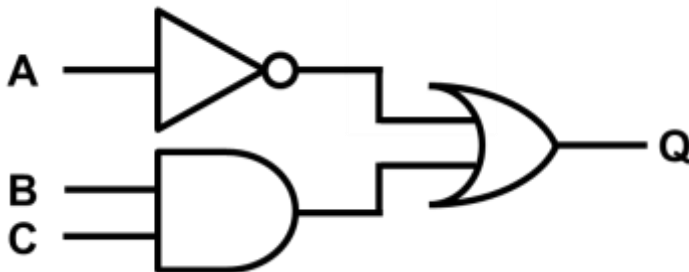
## 2.4 – Computational Logic

### 2.4 Computational Logic

- Binary is a base two number system where the two possible digits are 0 and 1.
- Data is represented in binary form in computer systems because computers are based on electrical circuits where we can detect whether current is flowing or not (only two possibilities, TRUE or FALSE).
- FALSE equates to 0 and TRUE equates to 1.

Logic gate	Diagram	Truth table		
<b>AND</b> (conjunction)		A	B	$A \wedge B$
		FALSE	FALSE	FALSE
		FALSE	TRUE	FALSE
		TRUE	FALSE	FALSE
		TRUE	TRUE	TRUE
<b>OR</b> (disjunction)		A	B	$A \vee B$
		FALSE	FALSE	FALSE
		FALSE	TRUE	TRUE
		TRUE	FALSE	TRUE
		TRUE	TRUE	TRUE
<b>NOT</b> (negation)		A	$\neg A$	
		FALSE	TRUE	
		TRUE	FALSE	

- Logic gates can be combined into logic circuits, e.g.



$$Q = (\neg A) \vee (B \wedge C)$$

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

A	B	C	$\neg A$	$B \wedge C$	Q
0	0	0	1	0	1
0	0	1	1	0	1
0	1	0	1	0	1
0	1	1	1	1	1
1	0	0	0	0	0
1	0	1	0	0	0
1	1	0	0	0	0
1	1	1	0	1	1

### 2.5 Translators and Facilities of Languages

Type of language	Examples	Characteristics	Purpose	Sample instructions
<b>High-level</b>	Python C++ Java Visual Basic	- portable - one statement translates into multiple machine code instructions - translated using compiler or interpreter	enables humans to code programs more efficiently and easily	rate = 3.02 hrs = 15 pay = rate * hrs
<b>Low-level</b>	Assembly language	- processor-specific - one statement translates into one machine code instruction - translated using assembler	represents machine code using mnemonics (easier to understand)	STO &39FC LDA #34 ADD &4F3A
	Machine code	- processor-specific - written in binary - produced by interpreter, compiler or assembler	executes instructions within CPU	0011111101010110 1010001110101011

- **Translators** are used to convert code between high and low level languages so as to allow humans to write high-level code and execute it at processor level.

Type of translator	Languages using this	Characteristics
<b>interpreter</b>	Python	- translates code line by line - produces machine code - no object code is produced - identifies errors as program runs
<b>compiler</b>	C++	- translates program completely - produces machine code - object code is produced - identifies errors before program runs
<b>assembler</b>	Windows MASM	- translates assembly language into machine code - every assembly language instruction translates into one machine code instruction

- Tools and facilities available in an Integrated Development Environment (IDE):
  - **editors:** includes auto-indent, auto-correct, keyword colouring and line numbers
  - **error diagnostics:** identifies errors found during compilation
  - **run-time environment:** allows user to run (and test) a program within the IDE
  - **translators:** interpreters and compilers are generally part of an IDE

## 2.6 – Data Representation

### 2.6 Data Representation

#### Units

Unit	Number of bytes
Kilobyte (KB)	$10^3$
Megabyte (MB)	$10^6$
Gigabyte (GB)	$10^9$
Terabyte (TB)	$10^{12}$
Petabyte (PB)	$10^{15}$

#### Note

When a calculation gives you a result of 1024 MB, for example, you can usually round this to 1GB.

- Data needs to be converted into a binary format to be processed by a computer because computers can only process TRUE and FALSE data (as an on/off switch).
- 8-bit binary digits can have a value between 0 and 255 inclusive (256 numbers in total).

**Numbers**

- Converting binary into denary:

Binary number	<b>0</b>	<b>1</b>	<b>1</b>	<b>0</b>	<b>1</b>	<b>1</b>	<b>0</b>	<b>1</b>
Denary equivalent (for this position)	128	64	32	16	8	4	2	1
Denary value (for this bit)	0	64	32	0	8	4	0	1
Total denary value	<b>109</b> (total of above row)							

- Converting denary into binary:

Denary number	<b>179</b>							
Denary equivalent (for each position)	128	64	32	16	8	4	2	1
Does the number fit into this?	Yes	No	Yes	Yes	No	No	Yes	Yes
If so, subtract number from total	51	51	19	3	3	3	1	0
Binary	<b>1</b>	<b>0</b>	<b>1</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>1</b>

- Adding binary numbers:

$$\begin{array}{r}
 \phantom{1} \phantom{0} \phantom{0} \phantom{1} \phantom{0} \phantom{1} \phantom{0} \phantom{0} \\
 \phantom{1} \phantom{0} \phantom{1} \phantom{0} \phantom{1} \phantom{1} \phantom{1} \phantom{0} \phantom{1} \\
 \hline
 1 \phantom{0} \phantom{0} \phantom{1} \phantom{0} \phantom{1} \phantom{0} \phantom{0} \phantom{0} \\
 0 \phantom{1} \phantom{0} \phantom{1} \phantom{1} \phantom{1} \phantom{1} \phantom{0} \phantom{1} \\
 \hline
 1 \phantom{1} \phantom{1} \phantom{1} \phantom{1} \phantom{0} \phantom{0} \phantom{0} \phantom{1}
 \end{array}
 +$$

**Note**

- Adding 0 and 0 → 0
- Adding 0 and 1 → 1
- Adding 1 and 1 → 0 with 1 carried over

- An **overflow error** is when there are not enough bits to represent the number, so a ninth bit is needed, e.g.:

$$\begin{array}{r}
 \phantom{1} \phantom{1} \phantom{1} \phantom{1} \phantom{1} \phantom{1} \phantom{1} \phantom{1} \\
 \phantom{1} \phantom{1} \phantom{1} \phantom{0} \phantom{1} \phantom{0} \phantom{1} \phantom{0} \phantom{1} \\
 \phantom{1} \phantom{0} \phantom{1} \phantom{0} \phantom{1} \phantom{1} \phantom{1} \phantom{1} \phantom{1} \\
 \hline
 1 \phantom{0} \phantom{0} \phantom{1} \phantom{1} \phantom{0} \phantom{1} \phantom{0} \phantom{0}
 \end{array}
 +$$

**Note**

If you get a question that asks for an 8-bit answer but ends up causing an overflow error, make sure you only include the eight bits and **not** the ninth.

- A binary shift can be performed on a binary number to either multiply or divide it by 2:
  - moving to the left multiplies the number by 2
  - moving to the right divides the number by 2
- E.g. 0000 1111 after a binary shift left of two places becomes 0011 1100 (multiplied by 4)
- E.g. 1110 1100 after a binary shift right of three places becomes 00011101 (divided by 6)

## 2.6 – Data Representation

- Converting denary to hexadecimal: Hexadecimal (0 to 16) digits:

Denary number, n	<b>156</b>
$n \div 16$	9 remainder 12
Values of two hex digits	9 and 12
Hex	<b>9C</b>

<b>0</b>	0	<b>4</b>	4	<b>8</b>	8	<b>12</b>	C
<b>1</b>	1	<b>5</b>	5	<b>9</b>	9	<b>13</b>	D
<b>2</b>	2	<b>6</b>	6	<b>10</b>	A	<b>14</b>	E
<b>3</b>	3	<b>7</b>	7	<b>11</b>	B	<b>15</b>	F

- Converting hexadecimal to denary:

Hexadecimal	<b>8</b>	<b>F</b>
Denary equivalent	$8 \times 16$ $= 128$	15
Denary	<b>153</b>	

### Characters

- A **character set** is a list of characters and their codes recognised by the computer hardware and software.

- E.g. the character 'A' has:
  - denary code: 65
  - binary code: 1000 0001
- 'B' will have the next code
  - ('B' therefore has 66)
  - ('B' therefore has 1000 0010)

#### Note

Characters **are** case-sensitive, and every character (including special characters and numbers) has a unique character code.

- As the number of bits per character is increased, the number of characters which can be represented by a character set increases:

Character set	Bits per character, n	Number of characters, $2^n$
ASCII	7	128
Extended ASCII	8	256
Unicode	16	65 536

### Images

- An image is represented as a series of pixels (small squares), each pixel's colour represented by a binary number.
- An image file will store the colour of each pixel (in binary) in a specific order so that the picture can be recreated when a user wants to view it.
- The size of an image is represented by its width x height (in pixels), e.g. 400 x 600.
- **Metadata** is a set of data about another set of data, e.g. for an image file:
  - file type
  - time and date of creation
  - time and date of last modification
  - author
  - file size
  - resolution (total number of pixels)
  - dimensions of image
  - colour depth **number of colours =  $2^{\text{colour depth}}$**
- As colour depth increases, file size increases as more bits are used to represent each pixel.
- As resolution increases, file size increases because more bits are used altogether.

### Sound

- How an analogue signal is converted into digital format:
  - analogue sound received by microphone
  - microphone converts sound into electrical analogue signal
  - amplitude is measured at regular intervals (sampling)
  - values are quantised (rounded)
  - values stored as binary numbers
- Factors affecting sound file size and quality:
  - **sample rate:** number of audio samples per second, in Hertz (Hz)
  - **duration:** how many seconds of audio the sound file contains
  - **bit depth:** number of bits available to store each sample

### Compression

- **Compression** is the process of encoding data so that it needs fewer bits to represent it, to:
  - enable transmission over the Internet
  - enable faster transmission (at a given bandwidth)
  - free up space
- Lossy compression removes some accuracy from files, e.g. images, to make smaller files.  
+ decreases file size      – no good where 100% accuracy required, e.g. text files
- Lossless compression reorders data in files more efficiently to reduce file size.  
+ reversible + decreases file size      – cannot compress text-based files











# **COMPUTER SCIENCE**

## **PAPER 1 – COMPUTER SYSTEMS**

- 
- 1 SYSTEMS ARCHITECTURE

---

  - 2 MEMORY

---

  - 3 STORAGE

---

  - 4 WIRED AND WIRELESS NETWORKS

---

  - 5 NETWORK TOPOLOGIES, PROTOCOLS AND LAYERS

---

  - 6 SYSTEM SECURITY

---

  - 7 SYSTEM SOFTWARE

---

  - 8 ETHICAL, LEGAL, CULTURAL AND ENVIRONMENTAL CONCERNS
- 

## **PAPER 2 – COMPUTATIONAL THINKING, ALGORITHMS AND PROGRAMMING**

- 
- 1 ALGORITHMS

---

  - 2 PROGRAMMING TECHNIQUES

---

  - 3 PRODUCING ROBUST PROGRAMS

---

  - 4 COMPUTATIONAL LOGIC

---

  - 5 TRANSLATORS AND FACILITIES OF LANGUAGE

---

  - 6 DATA REPRESENTATION
-