Year 11 Curriculum Overview Computer Science 2023-24

	Term 1				
Unit Title	Compression	Networks and security	Systems software	Ethical, Legal, Cultural and Environmental (ELCE) impacts	Search and sort algorithms revision
Approximate Number of Lessons	1	14	3	7	6
Curriculum Content ***Note: Where appropriate, lessons will also include programming tasks***	Students will be able to describe types of compression and how they can be used on data	Students will study how networks are created and their uses. They will also learn how to protect computer systems from attack.	Students will learn about the software needed to manage and run a computer.	Students will learn about the positive and negative impacts of technology as well as the laws that govern its use.	Students will study how to read and write the standard search and sort algorithms.
Links to prior learning	Units of storage, Data storage	Systems architecture Memory and storage Programs	System architecture Memory and storage Networks and security	Networks and security	Strong links to writing and understanding algorithms
Cultural Capital Opportunities	Watch Wreck it Ralph: Breaks the internet	Watch Wreck it Ralph: Breaks the internet, The imitation game Take part in the Cyber Discovery challenge: https://joincyberdiscovery.com/ Visit Computing history centre in Cambridge or the National Museum of computing at Bletchley Park	Improve your computer's performance by running a disk clean up and defragmentation.	Watch/ read Ready Player One, Watch The Circle, The Social network Read articles from BBC technology: Technology - BBC News	Take part in the Cyber Discovery challenge: https://joincyberdiscovery.com/ Learn a new programming language
Assessment Focus	One 60 mark, 1 hour assessment focusing on all topics up to this point including year 9 and 10 split into component 1 and component 2 One full mock consisting of one paper on Computer systems and a second paper on Algorithms and programming				
Name of Knowledge Organiser	1.2 Units of storage and compression	1.3.1 Networks and topologies 1.3.2 Wired and wireless networks, Protocols and Layers 1.4 Network Security	1.5 Systems Software	1.6 Ethical, Legal, Cultural and Environmental impacts	2.1 Algorithms

Year 11 Curriculum Overview Computer Science 2023-24

	Term 2					
Unit Title	Algorithms and programming revision	Systems architecture revision	Memory and storage revision	Boolean logic revision	Networks and security revision	Ethical, Legal, Cultural and Environmental (ELCE) impacts revision
Approx Number of Lessons	5	2	5	2	3	2
Curriculum Content *Note: Where appropriate, lessons will also include programming tasks*	Students will recap how to read and write algorithms. Physical computing will be used to help students engage with the tasks	Students will recap the CPU and how it is used with other components of a computer.	Students will recap how data is stored by computers. This will include the devices data is stored on and how each type of data can be represented in binary.	Students will recap the rules of logic gates and how to apply them in truth tables	Students will recap how networks are created and their uses. They will also learn how to protect computer systems from attack.	Students will recap the positive and negative impacts of technology as well as the laws that govern its use.
Links to prior learning	Programming techniques Algorithms Boolean logic	Systems architecture Memory and storage Programs	Systems architecture Memory and storage Programs Boolean logic Basic numeracy skills	Previous learning on Boolean logic and Boolean operators	Systems architecture Memory and storage Programs	Networks and security ELCE impacts
Cultural Capital Opportunities	Take part in the Cyber Discovery challenge: https://joincyberdiscovery.com/ Learn a new programming language	Visit Computing history centre in Cambridge or the National museum of computing at Bletchley park Watch Tron	Watch The Emoji movie, The Martian, Tron, Calculating Ada Visit Computing history centre in Cambridge or the National museum of computing at Bletchley park	Visit Computing history centre in Cambridge or the National museum of computing at Bletchley park Creating logic circuits on logic.ly/demo	Take part in the Cyber Discovery challenge Watch Wreck it Ralph: Breaks the internet, The imitation game Exploring the Internet of Things	Watch/ read Ready Player One, Watch The Circle, The Social network Read articles from BBC technology: Technology - BBC News
Assessment Focus	One 60 mark, 1 hour assessment focusing on all topics up to this point including year 9 and 10 split into component 1 and component 2 One full mock consisting of one paper on Computer systems and a second paper on Algorithms and programming					
Name of Knowledge Organiser	2.1 Algorithms 2.2 Programming Fundamentals 2.3 Producing Robust Programs		1.2 Memory and Storage 1.2 Number representation 1.2 Units of storage and compression 1.2 Images, Text and Sounds	2.4 Boolean logic	1.3.1 Networks and topologies 1.3.2 Wired and wireless networks, Protocols and Layers 1.4 Network Security	1.6 Ethical, Legal, Cultural and Environmental Impacts of digital technology

Year 11 Curriculum Overview Computer Science 2023-24

	Term 3
Unit Title	General revision
Approximate Number of Lessons	9
Curriculum Content ***Note: Where appropriate, lessons will also include programming tasks***	Students will recap all topics.
Links to prior learning	All topics
Cultural Capital Opportunities	Visit Computing history centre in Cambridge or the National museum of computing at Bletchley park
Assessment Focus	Component 1 final exam Component 2 final exam
Name of Knowledge Organiser	All component 1 and component 2 knowledge organisers

Mildenhall College ACADEMY

1.2 Units of Storage and Compression Knowledge Organiser

Key learning

Units

- **Bit, nibble, byte, kilobyte, megabyte, gigabyte, terabyte, petabyte**
- **How data needs to be converted into a binary format to be processed by a computer**

Compression

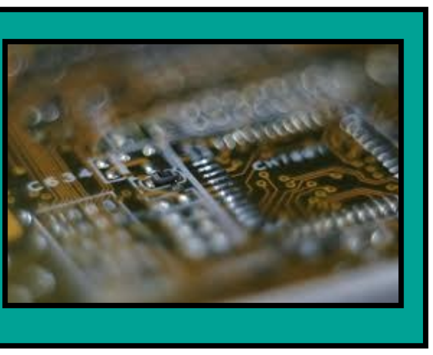
- **Need for compression**
- **Types of compression:**
  - **Lossy**
  - **Lossless**

Key terms

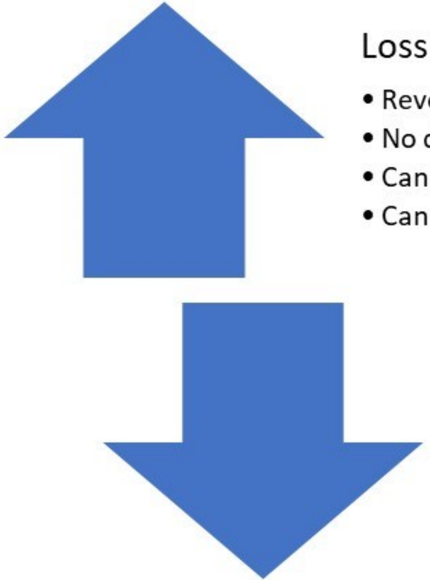
Bit	The smallest unit of data storage consisting of a single 1 or 0. This can be represented by a single transistor.
Nibble	A group of four bits (half a byte).
Byte	A group of 8 bits.
Compression	Reducing the file size to make it faster to send and take up less storage space.
Lossy	A method of compressing a file by permanently removing some data.
Lossless	A method of compressing a file keeping all of the data.

Why computers use binary

- Computers consist of many **transistors**
- Each **transistor** can only be **on** or **off**
- This can be used to represent **1** or **0**



Bit	Smallest unit of storage made of a sinlge 1 or 0
Nibble	A group of 4 bits
Byte	A group of 8 bits
Kilobyte	1 000 Bytes or 8 000 bits
Megabyte	1 000 Kilobytes or 1 000 000 Bytes
Gigabyte	1 000 Megabytes or 1 000 000 Kilobytes
Terabyte	1 000 Gigabytes or 1 000 000 Megabytes
Petabyte	1 000 Terabytes or 1 000 000 Terabytes



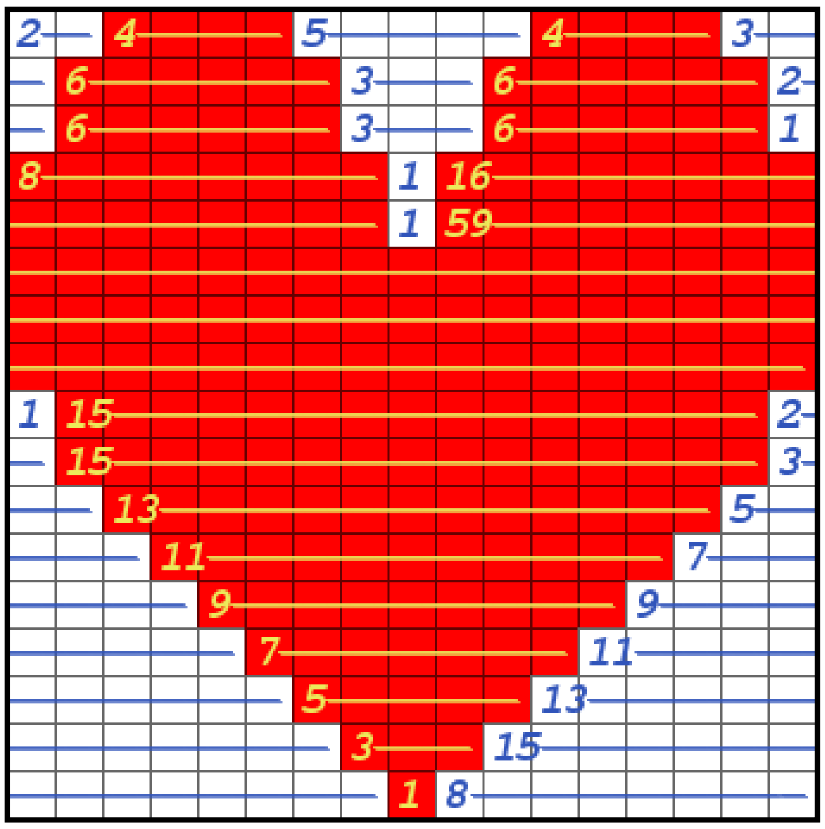
Lossless

- Reversible compression
- No data is lost
- Can be used on text
- Cannot be compressed as much as lossy

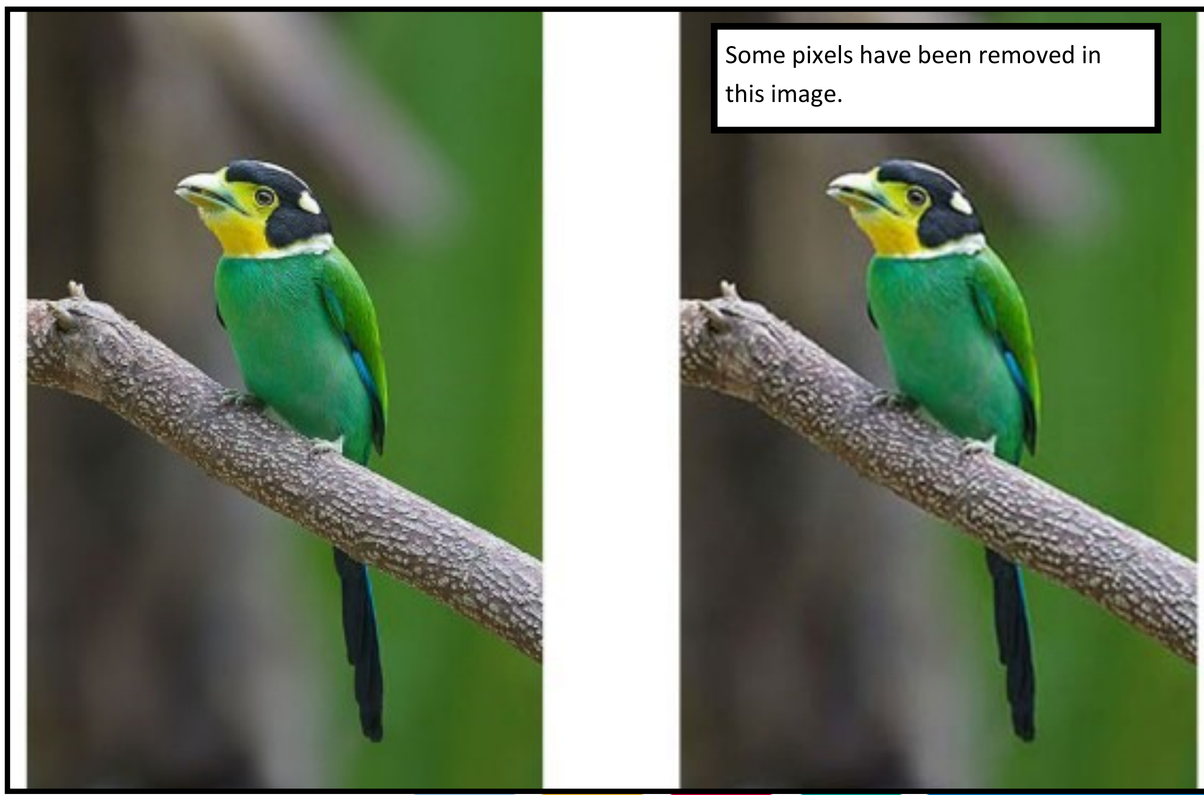
Lossy

- Permanent compression
- Some less innportant data will be lost
- Cannot be used on text
- Can be compressed more than lossless

Lossless compression



Lossy compression



Some pixels have been removed in this image.

Mildenhall College ACADEMY

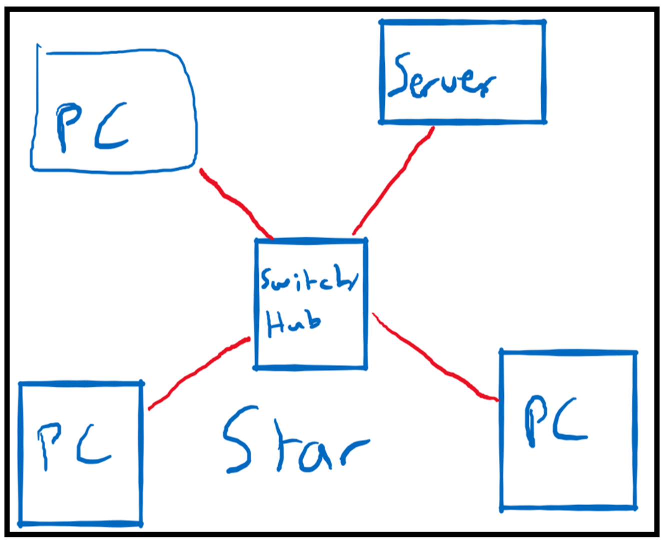
1.3.1 Networks and topologies Knowledge Organiser

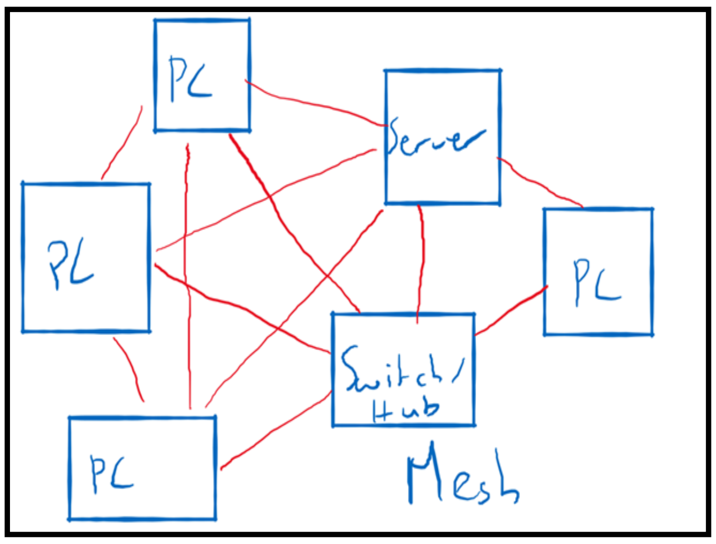
Key learning

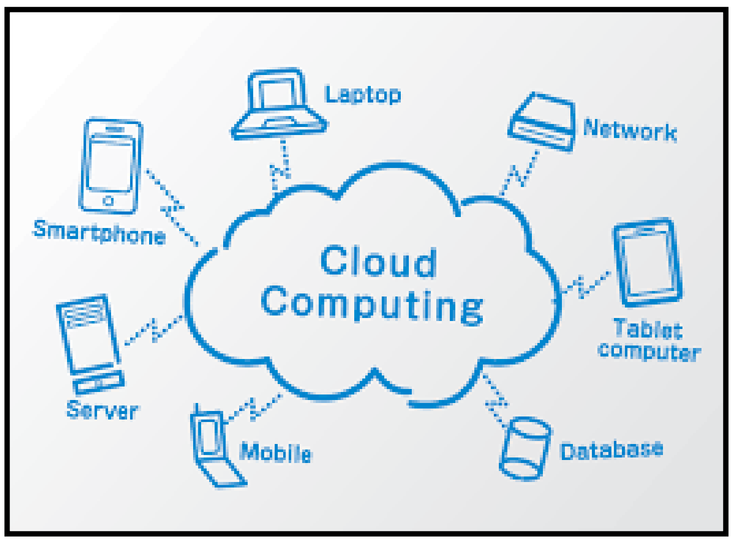
- Types of networks:
  - LAN (Local Area Network)
  - WAN (Wide Area Network)
- Factors that affect the performance of networks
- The different roles of computers in a client-server and a peer-to-peer network
- The hardware needed to connect stand-alone computers into a Local Area Network:
  - Wireless access points
  - Routers/ Switches
  - NIC (Network Interface Controller/Card)
  - Transmission media
- The internet as a worldwide collection of computer networks:
  - DNS (Domain Name Server)
  - Hosting
  - The cloud
  - Web servers and clients
- Network topologies (Star and mesh)

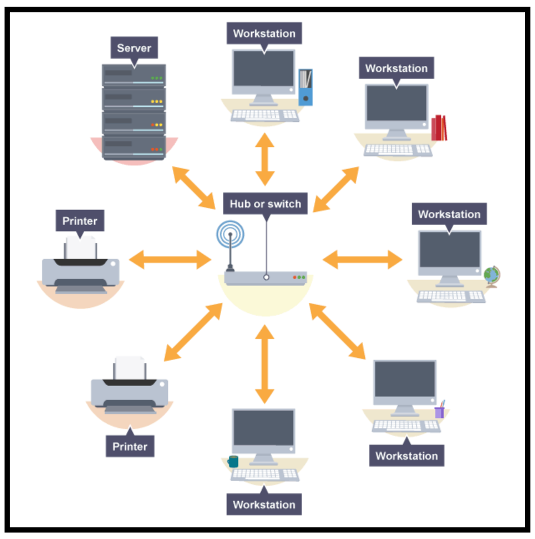
Key terms

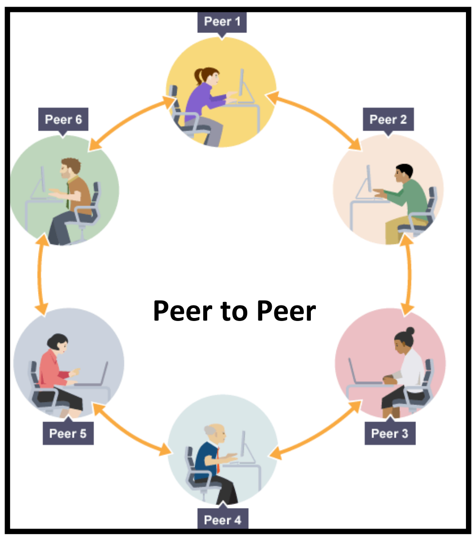
Network	2 or more computers connected together so they can communicate.
LAN	A connection of computers over a small distance, usually less than a mile.
WAN	A connection of computers over a large distance, usually more than a mile.
Client	A computer that requests data and files from a server.
Server	A computer which stores files and settings centrally and sends them to clients.
Peer	A computer with equal privileges.
Internet	The connection of computers and networks creating a world wide network.
WWW	The files and data stored on the internet.
Cloud	Using servers over the internet to store and manage files and data.
Topology	This is a way of setting up a LAN

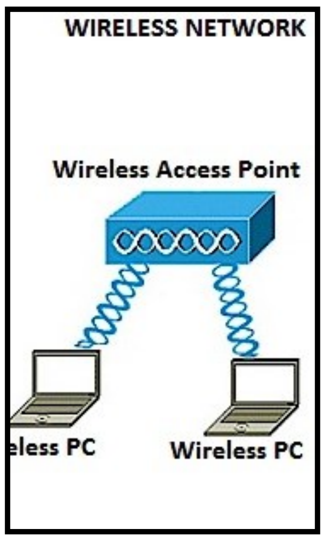


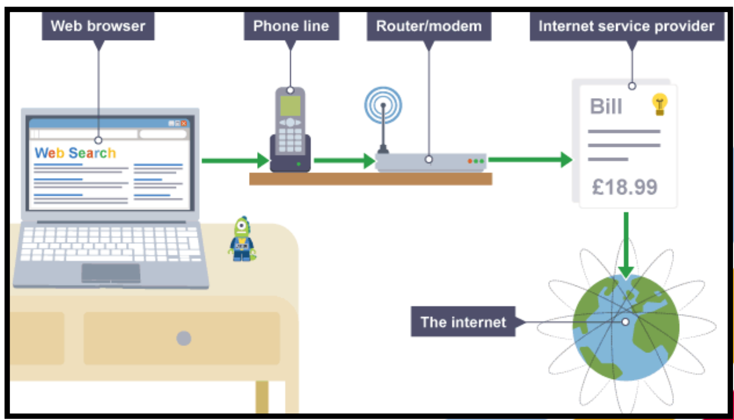


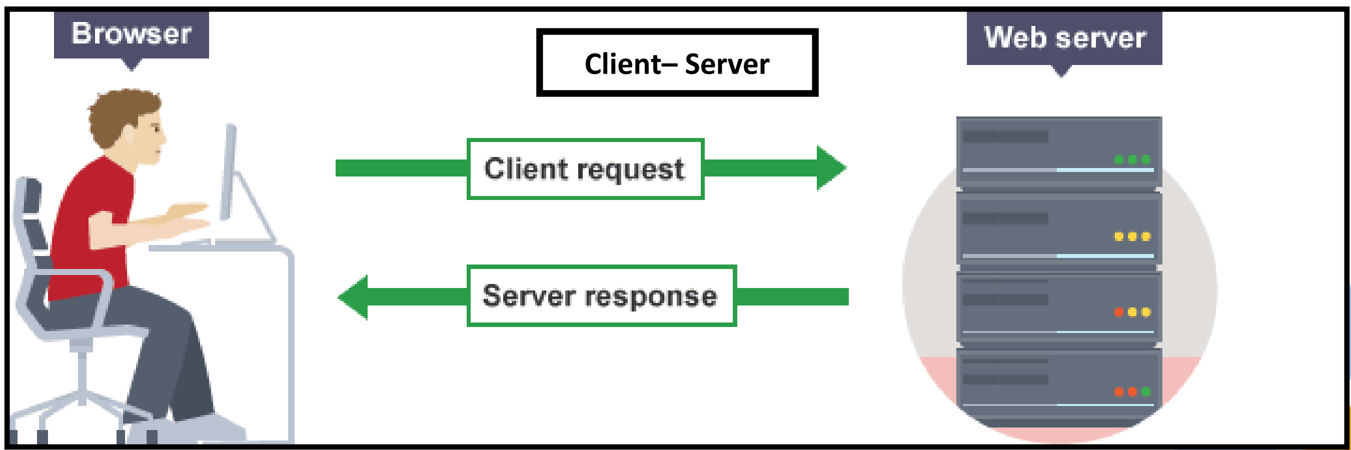


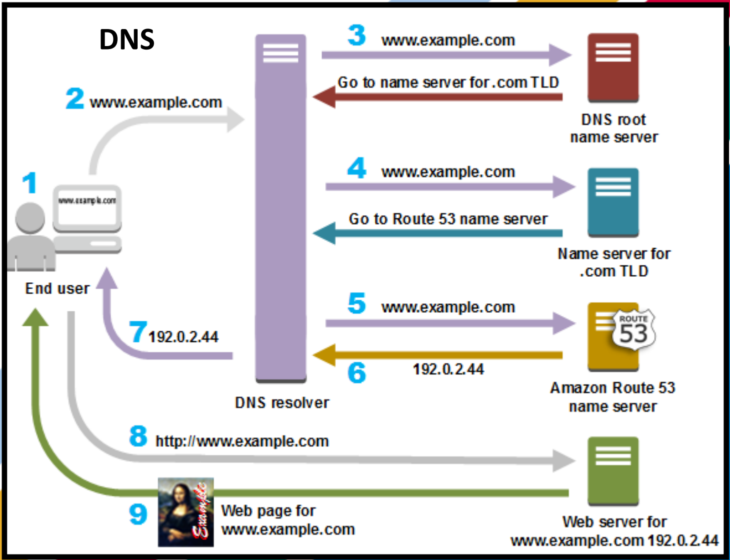








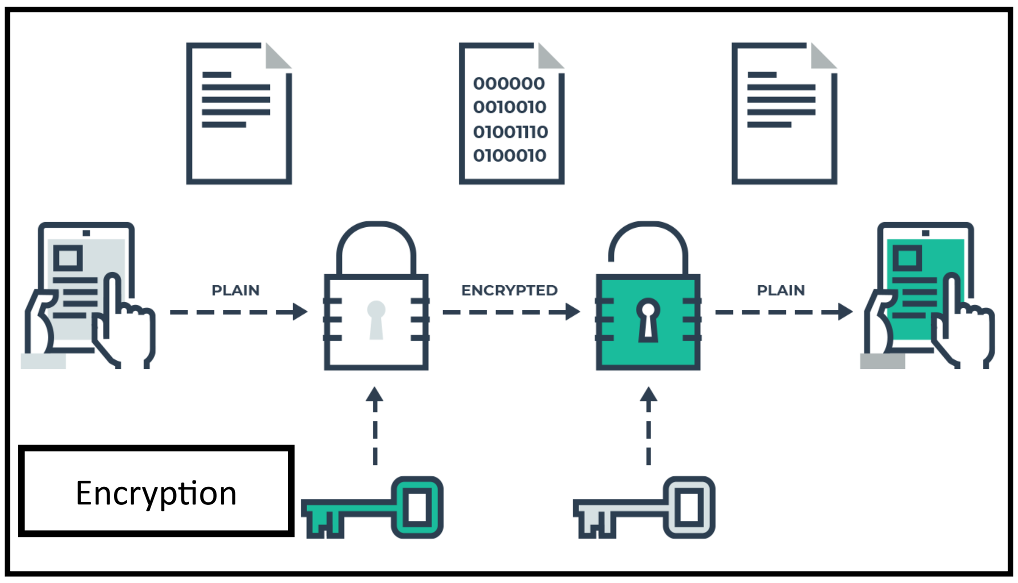




1.3.2 Wired and wireless networks, Protocols and Layers Knowledge Organiser

Key learning

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

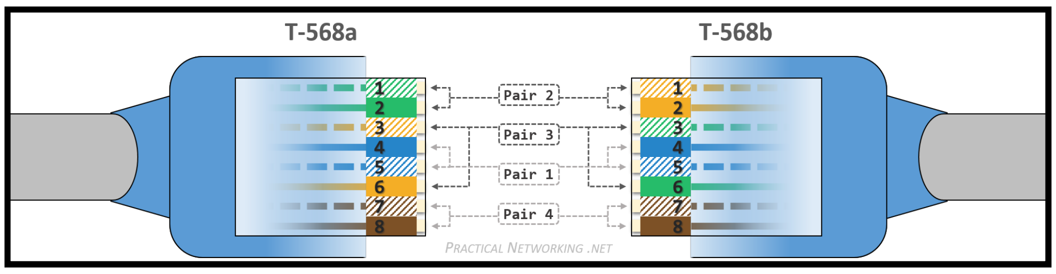


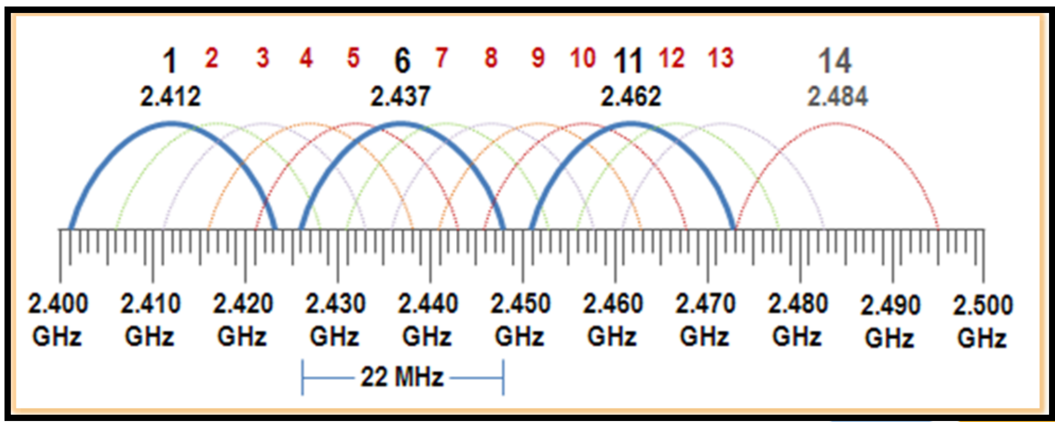
Encryption

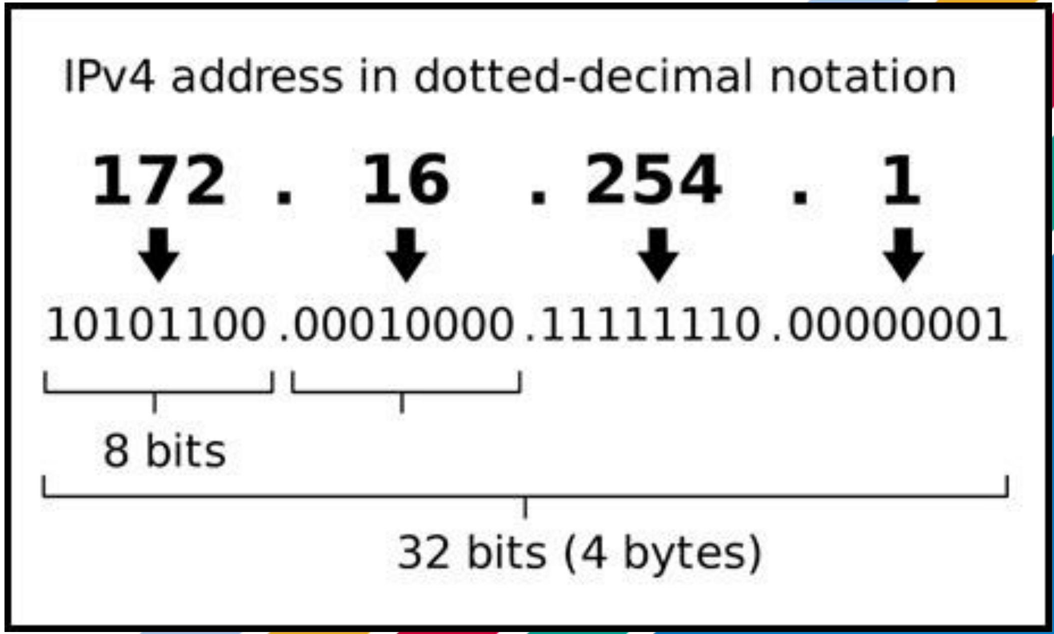
Application layer	Responsible for encoding or decoding the message/ file.
Transport layer	Breaks the message into small packets and gives each packet a number and total number of packets. This is used by recipient to put the message back together.
Network layer	Adds the IP address of sender and recipient to each packet. This allows the packets to get to the right destination.
Data link layer	This converts the packets into the signals to send over the network, either electrical pulses, light pulses or radio waves.

Key terms

Ethernet	The protocol used to send data through wires over a network.
Wi-Fi	A radio technology that allows devices to communicate without wires.
Bluetooth	A short range radio technology allowing transfer of data between devices
Protocol	A set of rules which govern how devices communicate to each other.
Encryption	Applying a key to plain text to convert it to something people will not understand (ciphertext)
Standards	Common protocols followed by different manufacturers to allow cross communication
Layers	A method for separating the different jobs of communicating over a network so that everything does not need to be changed to change one part.
IP addressing	A series of numbers/ values to identify devices on a network. Used to send data between networks
MAC addressing	A series of hexadecimal values used to uniquely identify a device. Used to send data within a network






IPv4 address in dotted-decimal notation

172 . 16 . 254 . 1

10101100.00010000.11111110.00000001

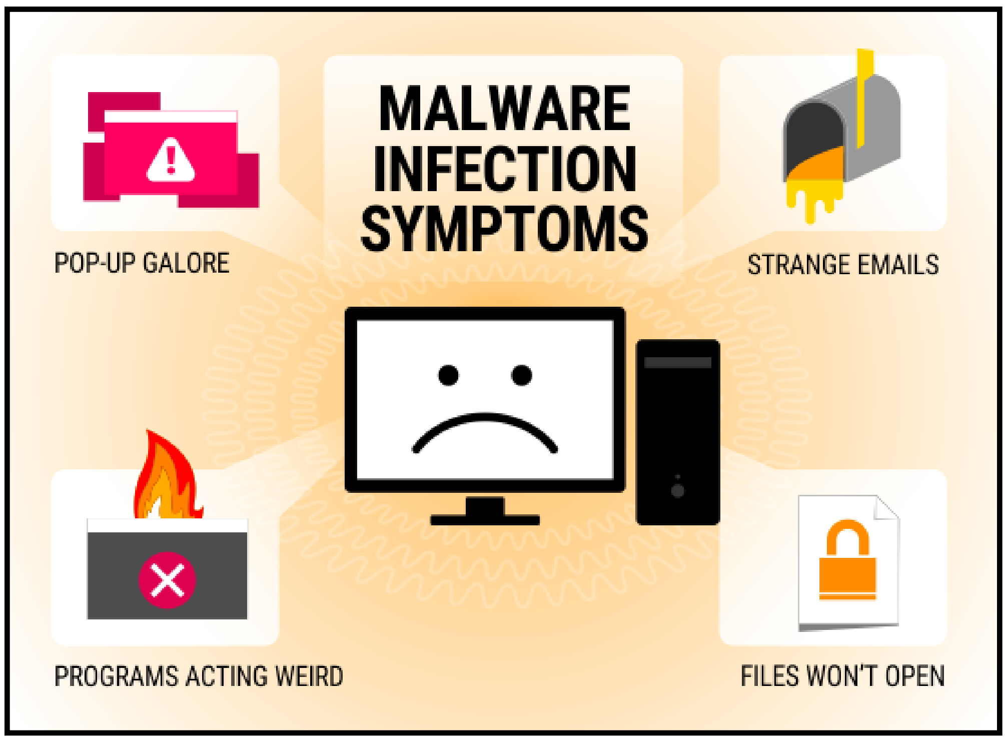
8 bits

32 bits (4 bytes)

1.4 Network Security Knowledge Organiser

Key learning

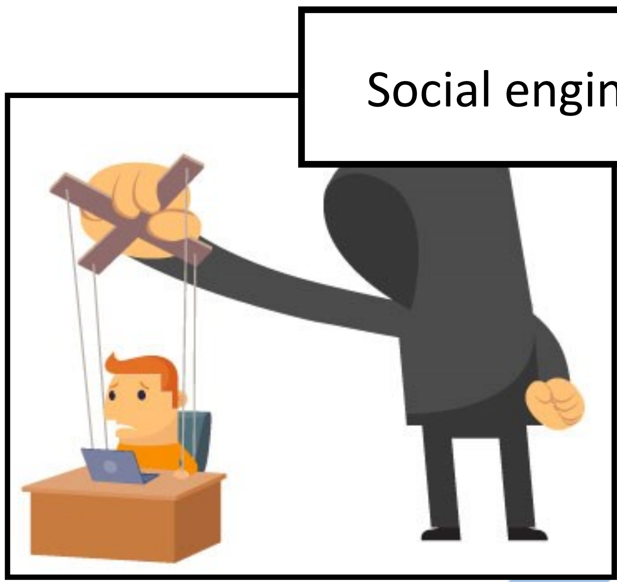
- **Forms of attack**
  - **Malware**
  - **Phishing**
  - **People as the ‘weak point’ in secure systems (social engineering)**
  - **Brute force attacks**
  - **Denial of service attacks**
  - **Data interception and theft**
  - **The concept of SQL injection**
- **Identifying and preventing vulnerabilities:**
  - **Penetration testing**
  - **Physical security**
  - **Anti-malware software**
  - **Firewalls**
  - **User access levels**
  - **Passwords**
  - **Encryption**

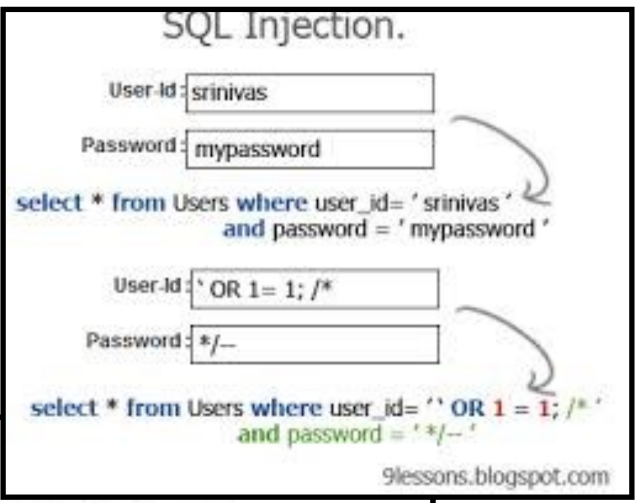


Key terms

Malware	Software designed to damage, delete, steal or copy files.
Phishing	Messages intended to trick people into thinking they are from a trusted person or company to get personal information.
Brute force	A type of attack where every possible combination is tried until the correct combination is entered.
Denial of Service	A type of attack where a single server is targeted by sending too much data for it to handle so that it crashes.
Data interception	A passive attack where the attacker listens in on the network to steal data being sent.
SQL injection	A type of attack that takes advantage of poorly written code to allow the attacker to add SQL to form fields to edit a database table.
Social engineering	Where an attacker will use deception to gain the trust of and manipulate individuals to gain information.
Firewall	A form of defence which blocks some of the ports and prevents data getting in that shouldn’t. Can be hardware or software.
Penetration testing	This is used to test the security of a system to reduce the chance of attackers gaining entry.
User access levels	A method for preventing users from accessing data that they do not need access to.

Social engineering



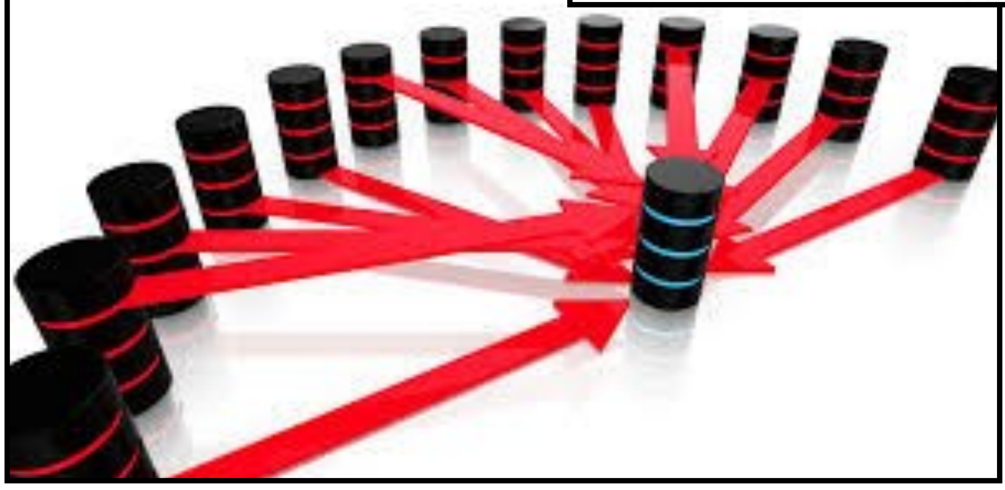


SQL Injection.

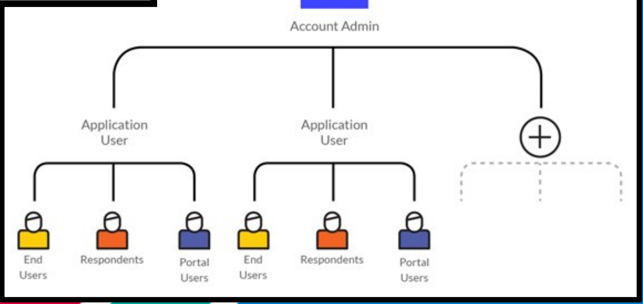
```
User-Id : srinivas
Password : mypassword

select * from Users where user_id= ' srinivas '
                   and password = ' mypassword '

User-Id : ' OR 1= 1; /*
Password : */--

select * from Users where user_id= '' OR 1 = 1; /* '
                   and password = ' */-- '
```

9lessons.blogspot.com

Denial of Service



Levels of Access



Mildenhall College
ACADEMY

1.5 Systems Software Knowledge Organiser

Key learning

- The purpose and functionality of systems software
- Operating systems:
  - File management
  - Resource management
  - User interface
  - Input/ output control
  - Task management
  - Security
- Utility system software:
  - Encryption software
  - Defragmentation
  - Data compression
  - Backing up

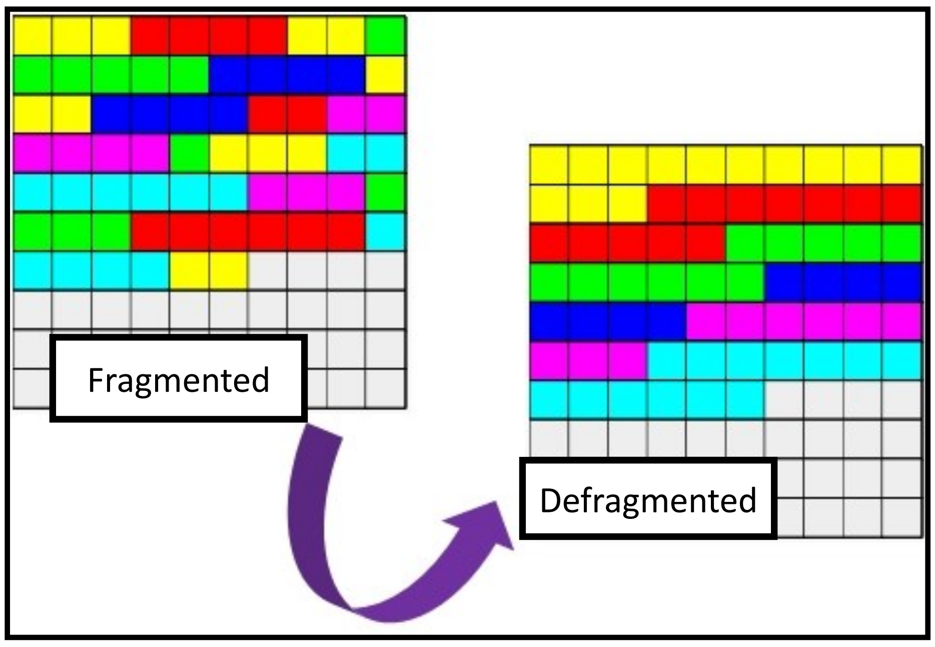
Key terms

Operating system	The piece of software which acts as the link between humans and the hardware.
Utility	This is software designed to help manage a computer. It will do one or two jobs really well.
Fragmentation	This is where gaps are created between files on a hard drive so it takes longer to retrieve what is needed.
Compression	File size is reduced either by grouping together data or removing dat. See 2.6.
Back up	A copy of files needed in case there is a failure of a computer system.

Jobs of an Operating System

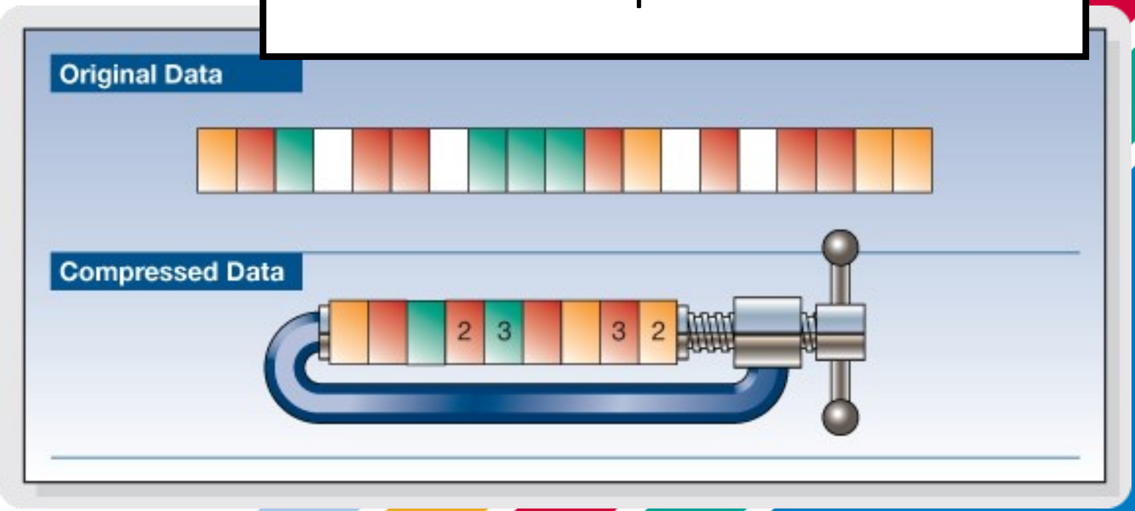
File management
Resource management
User interface
Input/ output control
Task management
Security

Defragmentation



Backing up

Compression



Mildenhall College
ACADEMY

1.6 Ethical, Legal, Cultural and Environmental Impacts of technology Knowledge Organiser

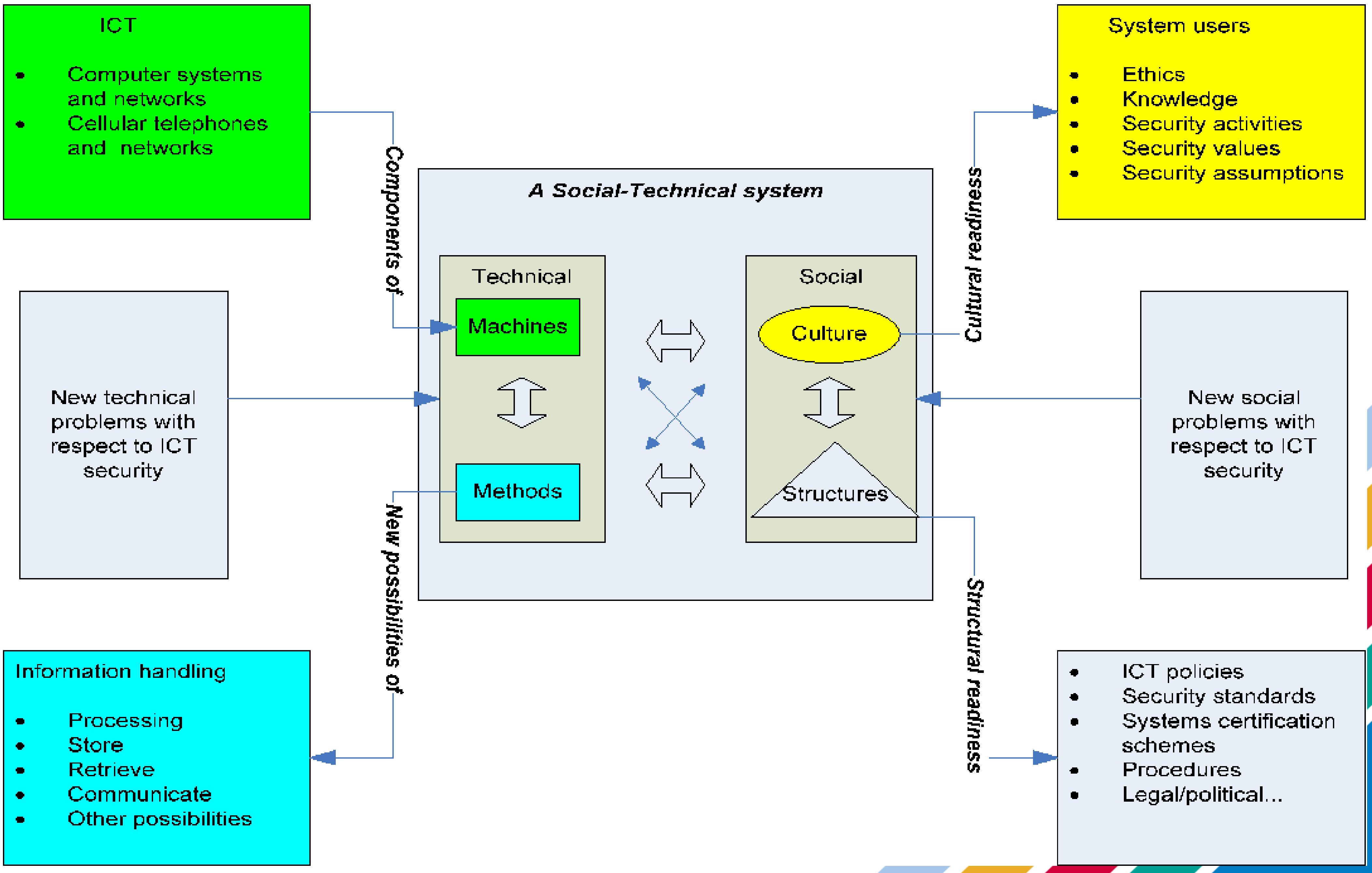
Key learning

- **How to investigate and discuss Computer Science technologies while considering:**
  - **Ethical issues**
  - **Legal issues**
  - **Cultural issues**
  - **Environmental issues**
  - **Privacy issues**
- **How key stakeholders are affected by technologies**
- **Environmental impact of Computer Science**
- **Cultural implications of Computer Science**
- **Open source vs proprietary software**
- **Legislation relevant to Computer Science:**
  - **The Data Protection Act 2018**
  - **Computer Misuse Act 1990**
  - **Copyright Designs and Patents Act 1988**

Keep an eye on what is going on in technology. The Technology section on BBC news is a good place to start.

Key terms

Ethics	These are the moral principles that dictate how a person behaves online and with technology.
Legislation	The laws that govern how computers and technology can be used.
Culture	How a person is brought up to behave when using technology and socialise using it.
Privacy	Being aware of what data should be shared online.
Open source	Software that can legally have the code edited to change how it behaves.
Proprietary software	Software where the code cannot be edited.
Stakeholder	A person or organisation who has an interest in the technology.



ICT
- Computer systems and networks
- Cellular telephones and networks

System users
- Ethics
- Knowledge
- Security activities
- Security values
- Security assumptions

A Social-Technical system

Technical: Machines, Methods

Social: Culture, Structures

New technical problems with respect to ICT security

New social problems with respect to ICT security

Information handling
- Processing
- Store
- Retrieve
- Communicate
- Other possibilities

- ICT policies
- Security standards
- Systems certification schemes
- Procedures
- Legal/political...

Mildenhall College
ACADEMY

1.1 CPU architecture, CPU performance and Embedded systems Knowledge Organiser

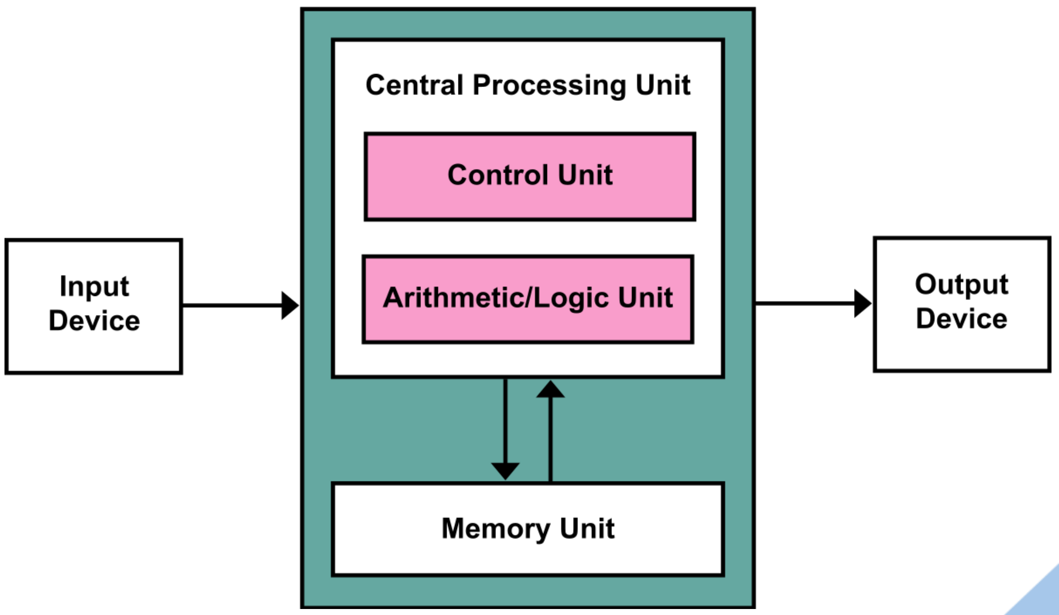
Key learning

- The purpose of the CPU
- Von Neumann architecture:
  - MAR (Memory Address Register)
  - MDR (Memory Data Register)
  - Program Counter
  - Accumulator
- Common CPU components and their function:
  - ALU (Arithmetic Logic Unit)
  - CU (Control Unit)
  - Cache
- The role of the fetch and execute cycle
- How common characteristics of CPUs affect their performance:
  - clock speed
  - cache size
  - number of cores
- Embedded systems:
  - purpose of embedded systems
  - examples of embedded systems

Key terms

CPU	The component responsible for executing instructions and processing data
Von Neumann	A type of design for a CPU
Register	A small data store on a CPU for a single piece of data (PC, MAR, MDR, ACC)
CU	The Control Unit is responsible for directing how to respond to instructions
ALU	The Arithmetic Logic Unit carries out the mathematical and logical operations
Fetch-execute cycle	The process of a CPU carrying out instructions stored in memory
Clock speed	The number of instructions processed per second
Cache	Fast memory, close to the CPU which stores frequently used instructions
Cores	Individual, sub processors on a CPU. Allows for multiple instructions to be executed at the same time.
Embedded system	A computer with a dedicated function built into an appliance.

Examples of Embedded Systems


Central Processing Unit

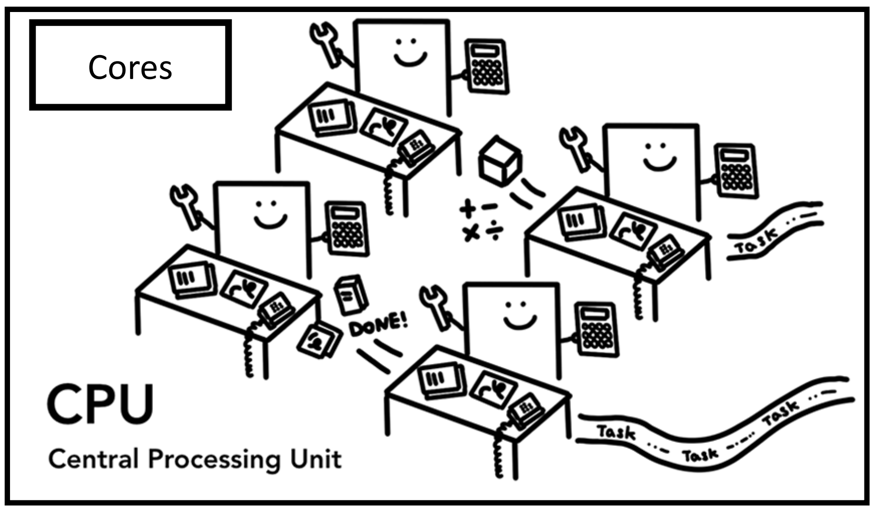
Control Unit

Arithmetic/Logic Unit

Memory Unit

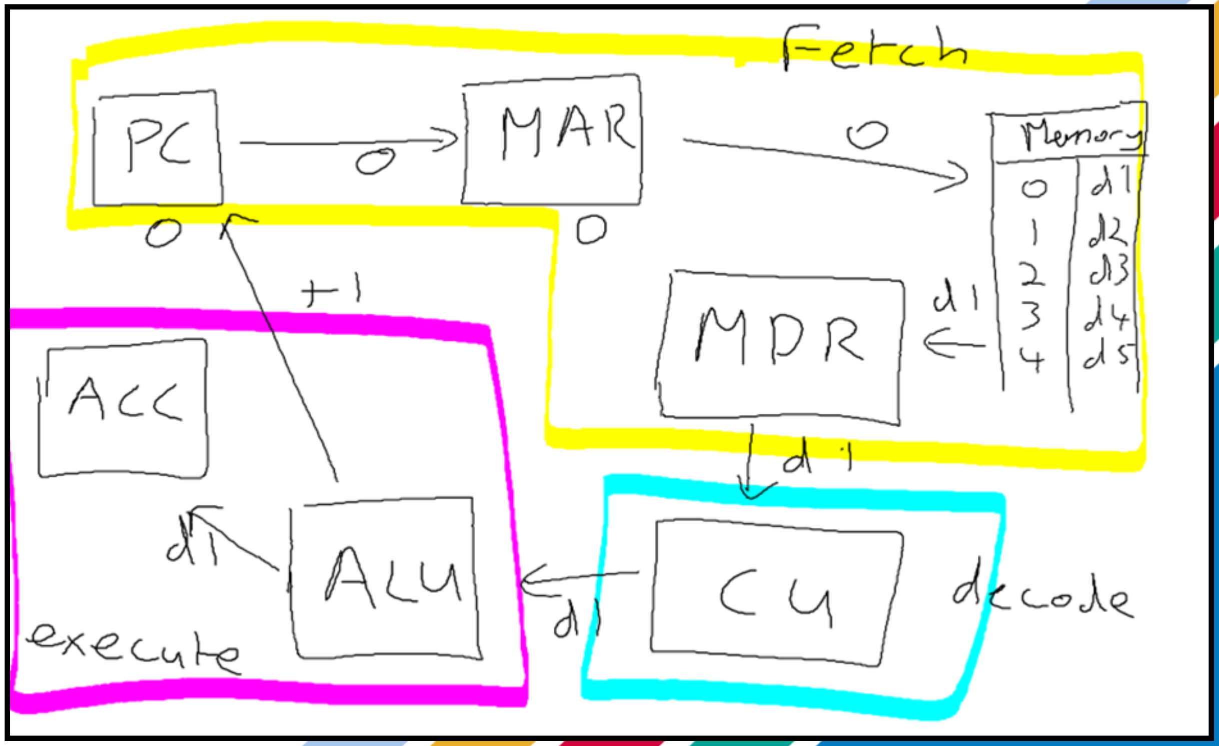
Input Device

Output Device


Cores

CPU

Central Processing Unit



Mildenhall College ACADEMY

1.2 Memory and Storage Knowledge Organiser

Key learning

1.2.1 Primary Storage (Memory)

- **The difference between RAM and ROM**
- **The purpose of ROM in a computer system**
- **The purpose of RAM in a computer system**
- **The need for virtual memory**

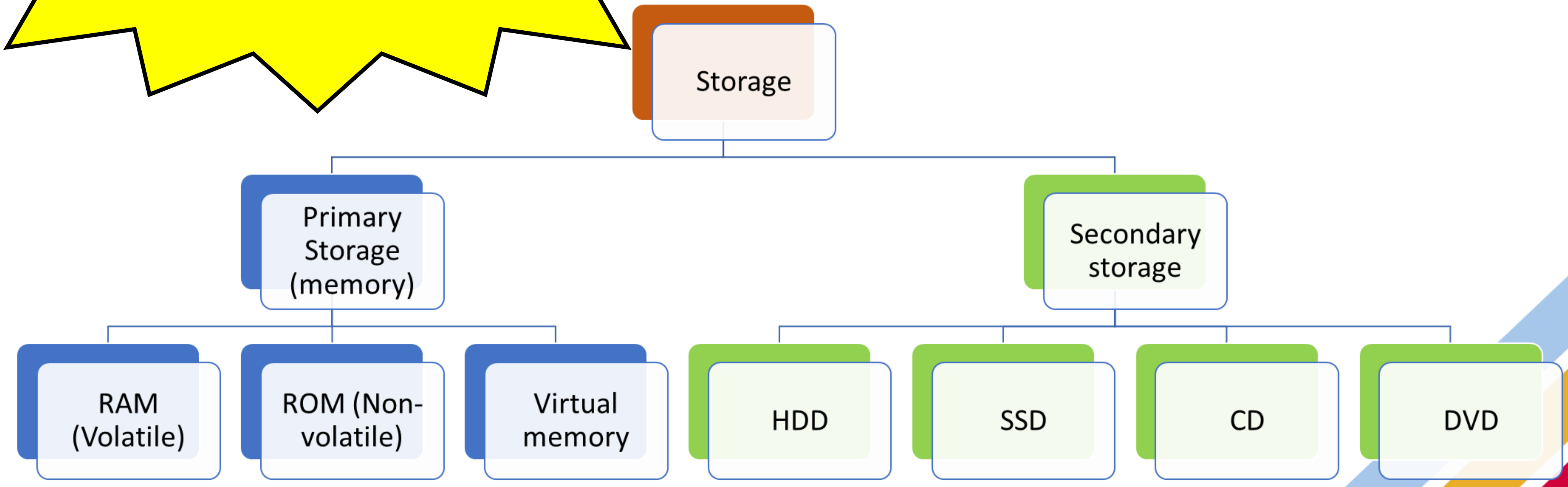
1.2.2 Secondary Storage

- **The need for secondary storage**
- **Data capacity and calculation of data capacity requirements**
- **Common types of storage:**
  - **Optical**
  - **Magnetic**
  - **Solid state**
- **Characteristics and suitable uses of storage devices:**
  - **Capacity**
  - **Speed**
  - **Portability**
  - **Durability**
  - **Reliability**
  - **Cost**

Key terms

Primary Storage (Memory)	A component which stores data and instructions for use by the CPU.
Secondary Storage	A component which stores files and data long term.
RAM	The computers working memory. It stores instructions and data whilst programs are running. This is volatile memory.
ROM	This is Read Only Memory normally used to store computer start-up instructions.
Virtual memory	A reserved part of the hard drive used like RAM.
Capacity	The amount of space available on a memory or storage device.
Magnetic	Data is stored using magnetic fields to represent 1s and 0s
Optical	Data is read using a laser to detect pits and falls on a CD/ DVD/ Blu-ray disc.
Solid state	Data is stored using electrical circuits with no moving parts.
Volatile	This is memory that will lose its data when power is lost.
Non-volatile	This is memory that doesn’t lose its data when power is lost.

Common misconception:
Secondary storage is only used as a back up store or extra storage



Storage

Primary Storage (memory)

RAM (Volatile)

ROM (Non-volatile)

Virtual memory

Secondary storage

HDD

SSD

CD

DVD

	Capacity	Speed	Portability	Durability	Reliability	Cost
Magnetic	High	Mid	Mid	Mid	Mid	Low
Optical	Low	Low	High	Mid	Mid	Mid
Solid state	Mid	High	High	High	High	High

Mildenhall College
ACADEMY

1.2 Number Representation Knowledge Organiser

Key learning

Numbers

- **How to convert positive denary whole numbers (0–255) into 8 bit binary numbers and vice versa**
- **How to add two 8 bit binary integers and explain overflow errors which may occur**
- **Binary shifts**
- **How to convert positive denary whole numbers (0–255) into 2 digit hexadecimal numbers and vice versa**
- **How to convert from binary to hexadecimal equivalents and vice versa**

Binary Addition

Rules

0 + 0 = 0

0 + 1 = 1

1 + 1 = 10

1 + 1 + 1 = 11

```
   0 1 1 0 0 1 1 1
+  1 0 0 1 1 1 0 1
 -----------------
 1 0 0 0 0 0 1 0 0
 -----------------
Carry 1 1 1 1 1 1 1 1
```

Binary shifts

Left shift

- Each **left shift** will **add one 0** to the **right hand side** of the binary number
- **Each shift doubles** the denary equivalent of the binary number
- If the **number exceeds maximum number** of bits then the **left digit is lost** this will **reduce the accuracy** of the number

Right shift

- Each **right shift** removes the **right hand digit** from the binary number
- **Each shift** will **divide** the denary equivalent of the number **by** 2
- If **1s** are removed then the **accuracy of the number is reduced**

Key terms

Bit	The smallest unit of data storage consisting of a single 1 or 0. This can be represented by a single transistor.
Nibble	A group of four bits (half a byte).
Byte	A group of 8 bits.
Binary	A base 2 system computers understand due to being made of transistors that can either be on or off.
Hexadecimal	A base 16 system used by humans to help remember and read binary code. Each binary nibble links directly to 1 hexadecimal digit.
Most significant bit	The left most digit of a binary number which has the highest value
Overflow error	When addition or left shifts lead to more than the original number of bits

Number conversions (Denary > Binary > Hex)

Binary to denary (01001101)

- Place the binary numbers under the **binary place values** starting from **right** to **left**
- **Add** together the headings **where there is a 1** underneath
- E.g. 64+8+4+1 =**77**

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

Denary to binary (56)

- Work from the **left** and attempt to **subtract** the heading from your number
- If you can do it without getting a negative number then put a 1 under the heading and use the answer in the next column
- If you can't put a 0 under the heading and move to the next column

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

Binary to hexadecimal (01001101)

- Split the **Byte** in half, this time use the top place values to convert each half (**nibble**) into **denary**
- If the **number is more than 9** use the letters **A to F** instead

A	10
B	11
C	12
D	13
E	14
F	15

E.G. the left would be 4, the right would be 8 + 4 + 1= 13

13 = D ➔ **Final answer = 4D**

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

Hexadecimal to Binary (F5)

- Use the **top headings** to convert each digit of the **hexadecimal** number to **binary**
- Make sure you keep them on the correct side (left to left and right to right)

F=15

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

Mildenhall College
ACADEMY

1.2 Images, Text and Sounds Knowledge Organiser

Key learning

Characters

- The use of binary codes to represent characters
- The term ‘character-set’
- The relationship between the number of bits per character in a character set and the number of characters which can be represented (for example ASCII, extended ASCII and Unicode)

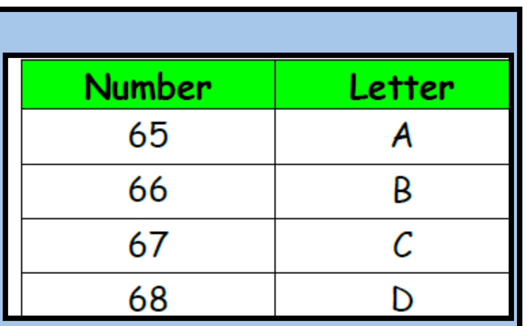
Images

- How an image is represented as a series of pixels represented in binary
- Metadata included in the file
- The effect of colour depth and resolution on the size of an image file

Sound

- How sound can be sampled and stored in digital form
- How sampling intervals and other factors affect the size of a sound file and the quality of its playback:
- Sample size
- Bit rate
- Sampling frequency

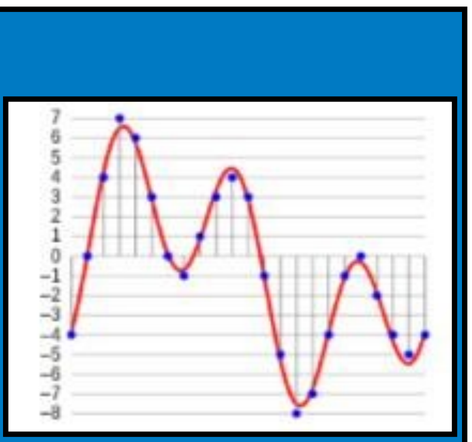
Text



Number	Letter
65	A
66	B
67	C
68	D

- Each **character** is given a **unique** number
- This is converted into binary
- Characters will always be in order, a, b, c, etc.
- A popular character set is **ASCII** which uses 8 bits per character
- ASCII can only store the Latin alphabet due to the **256** character limit
- **Unicode** is a character set which uses 16 or 32 bits per character
- **Unicode** includes **ASCII** as its first 256 characters
- **Unicode** then allows all other alphabets to be included, including emojis

Sounds

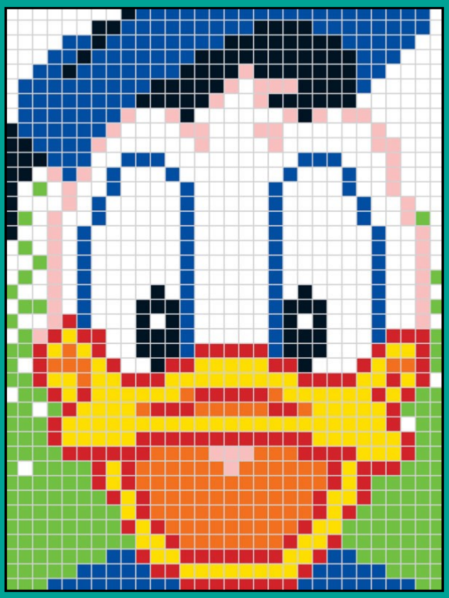


- Analogue sounds must be converted into digital sounds (binary)
- A **sample** is taken at **regular intervals** (**sample frequency**)
- A **sample** is a measurement of the amplitude at a set point in time
- Each **sample** is stored as a binary number
- The **accuracy of each sample** is determined by the **sample size**
- The **accuracy of the wave** is determined by the **sample frequency**
- Bit rate can be worked out by multiplying the **sample frequency** by the **sample size**
- File size can be worked out by multiplying the **bit rate** by the **length of the sound** in seconds

Key terms

Pixel	The smallest element of an image (picture element).
Resolution	The number of pixels in an image or defined area.
Colour depth	Number of bits used to represent a pixel. This affects the number of colours which can be represented.
Meta data	Data about a file such as date, file type, author, resolution, bit depth, etc.
Character Set	The range of symbols a computer understands.
ASCII	A character set using 8 bits per character storing the Latin alphabet.
Unicode	A character set using 16 or 32 bits allowing other languages to also be represented.
Digital sound	The result of a sound being sampled and stored on a computer in binary.
Analogue sound	The original sound before it is sampled by a computer.
Sample	Measuring the height/ amplitude of a sound wave at a specific point in time.
Sample rate	The number of samples recorded every second.
Bit depth/ sample size	The number of bits used to represent each sample.
Bit rate	The number of bits being processed every second. Worked out by multiplying the sample frequency by the sample size.

Images



- Each image is made up of **pixels**
- The pixel is stored as a **binary number** which represents the colour of the pixel
- Each colour has a **unique** binary number
- The number of colours is determined by the **colour depth**
- The number of pixels in an image is known as its **resolution** which can be worked out by multiplying the **width** and **height**
- Each image will also store **metadata** such as **file type**, **date taken**, **author**, **location**, etc.
- The **file size** of an image can be worked out by multiplying the **resolution** by the **colour depth**.

Mildenhall College
ACADEMY

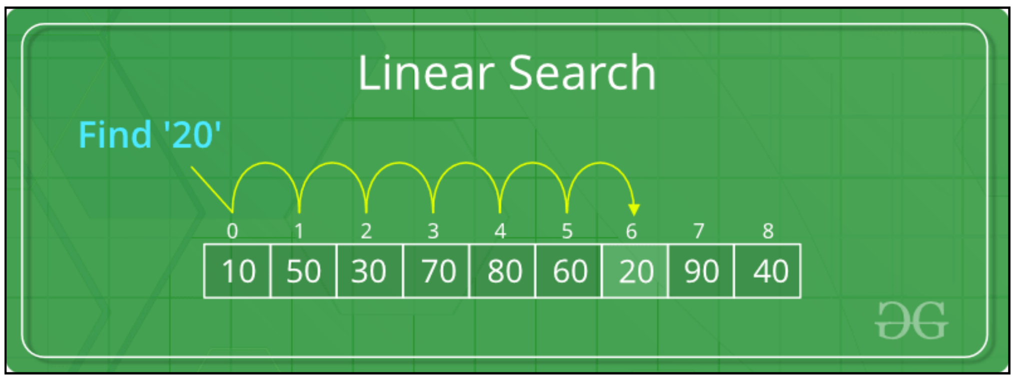
2.1 Algorithms Knowledge Organiser

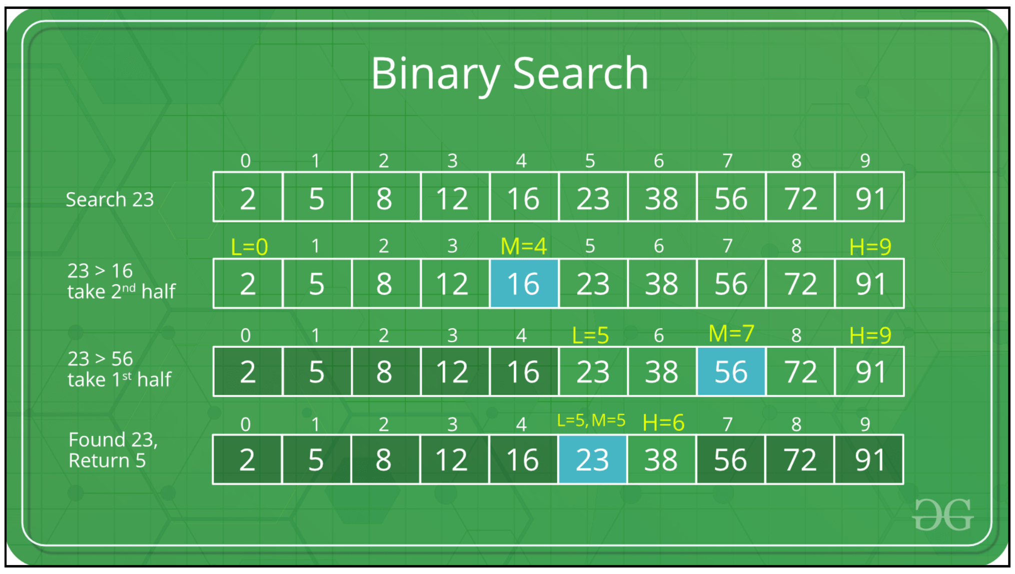
Key learning

- Computational thinking:
    - Abstraction
    - Decomposition
    - Algorithmic thinking
- Standard searching algorithms:
    - Binary search
    - Linear search
- Standard sorting algorithms:
    - Bubble sort
    - Merge sort
    - Insertion sort
- How to produce algorithms using:
    - Pseudocode
    - Using flow diagrams
    - Interpret, correct or complete algorithms

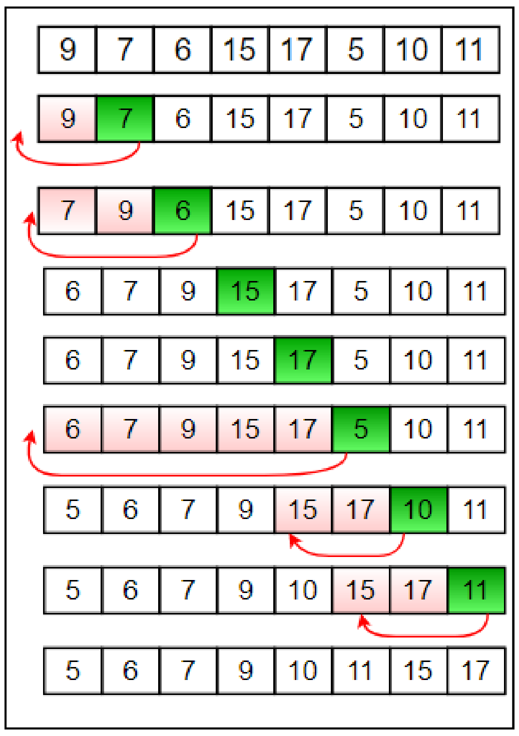
Key terms

Algorithm	A set of instructions to complete a task.
Abstraction	Removing unnecessary detail from a problem to make it easier to solve.
Decomposition	Breaking down a problem into smaller parts to make it easier to solve.
Algorithmic thinking	Identifying the steps needed to solve a problem.
Searching	An algorithm designed to find a piece of data in a list.
Sorting	An algorithm designed to sort a list into alphabetical or numerical order.
Pseudocode	A form of code which does not link to any programming language. It is used for planning.
Flow chart	A way of planning an algorithm using shapes to represent types of instruction.

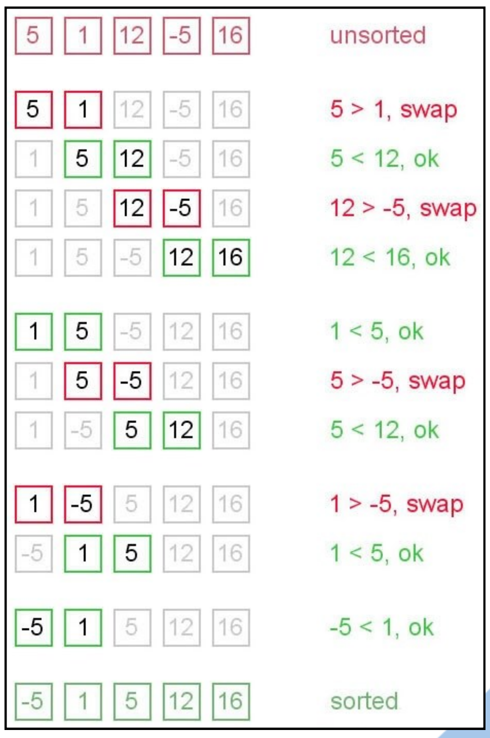


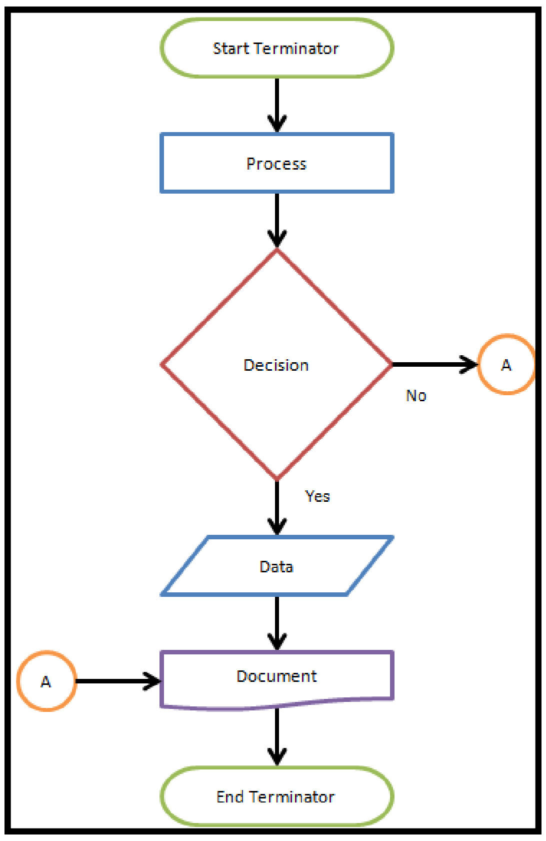


Insertion sort



Bubble sort

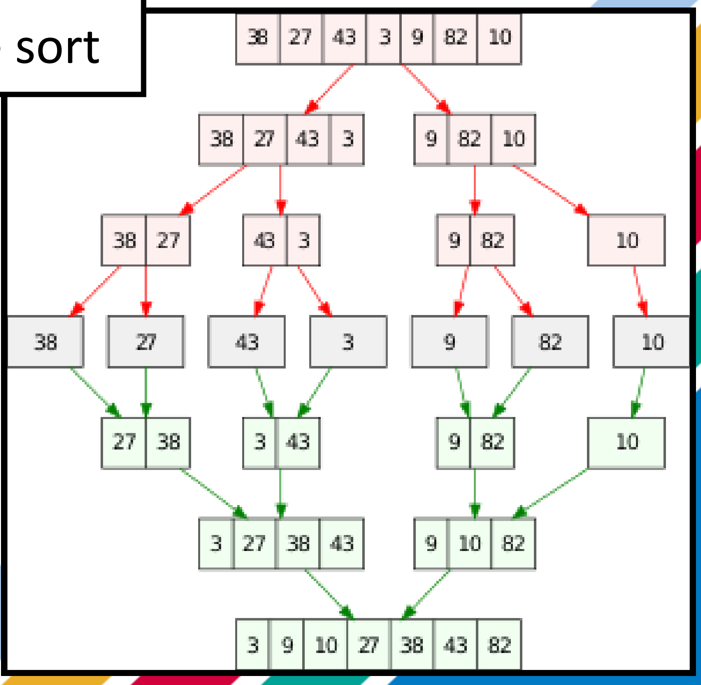


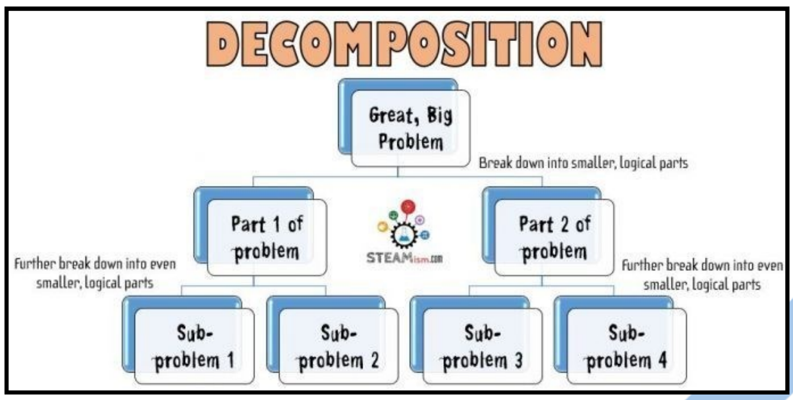


Pseudocode

- Use naming conventions
- Use indentation
- Make sure function names are clear
- Comment code

Merge sort

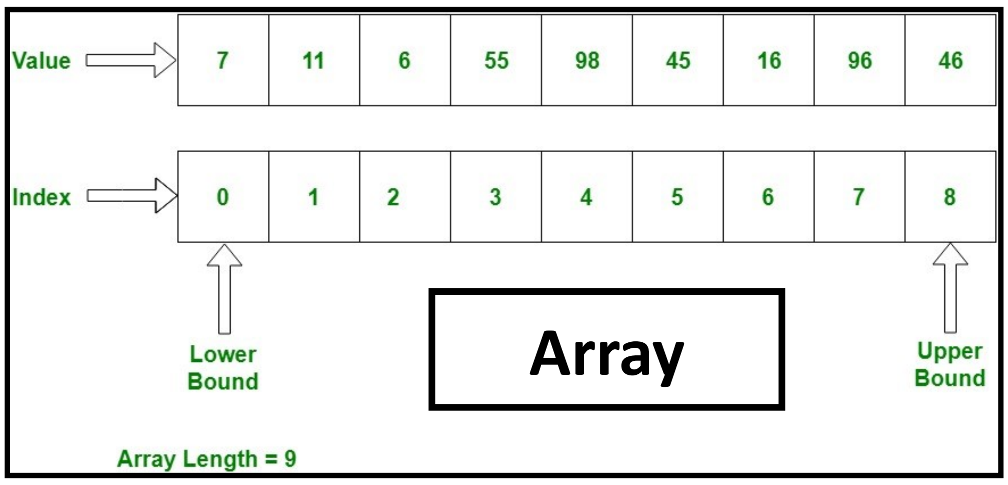



DECOMPOSITION
Great, Big Problem
Break down into smaller, logical parts
Part 1 of problem
Part 2 of problem
Further break down into even smaller, logical parts
Further break down into even smaller, logical parts
Sub-problem 1
Sub-problem 2
Sub-problem 3
Sub-problem 4

2.2 Programming Fundamentals Knowledge Organiser

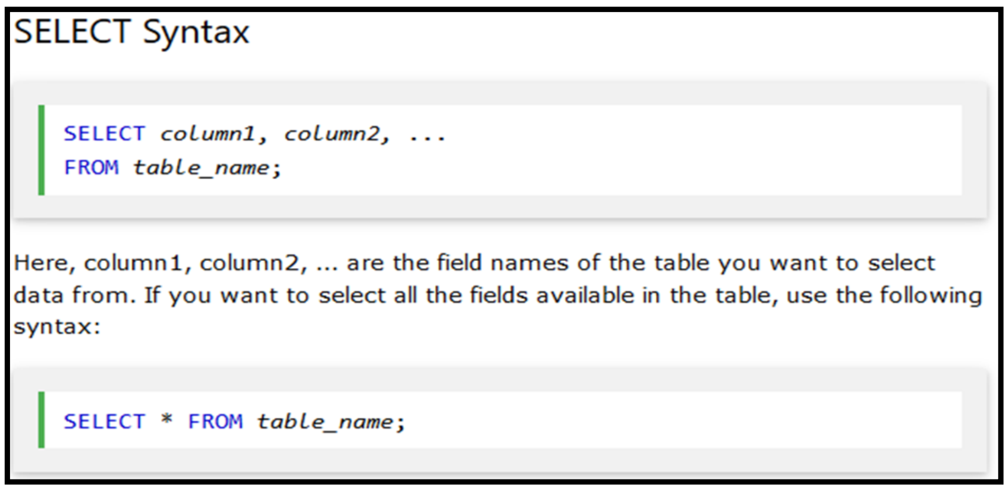
Key learning

- The use of variables, constants, operators, inputs, outputs and assignments
- The use of the three basic programming constructs used to control the flow of a program:
  - Sequence
  - Selection
  - Iteration (count and condition controlled loops)
- The use of basic string manipulation
- The use of basic file handling operations:
  - Open
  - Read
  - Write
  - Close
- The use of records to store data
- The use of SQL to search for data: SELECT, FROM, WHERE
- The use of arrays (or equivalent) when solving problems, including both one and two dimensional arrays
- How to use sub programs (functions and procedures) to produce structured code
- Random number generation
- The use of data types:
  - Integer
  - Real
  - Boolean
  - Character and string
  - Casting
- The common arithmetic operators: +, -, /, *, ^ MOD, DIV
- The common Boolean operators: AND, OR, NOT
- The common comparison operators: ==, !=, <, <=, >, >=



Value	7	11	6	55	98	45	16	96	46
Index	0	1	2	3	4	5	6	7	8

SQL example



SELECT Syntax

```
SELECT column1, column2, ...
FROM table_name;
```

Here, column1, column2, ... are the field names of the table you want to select data from. If you want to select all the fields available in the table, use the following syntax:

```
SELECT * FROM table_name;
```

Key terms

Variable	A named location in memory storing a single piece of data that can change.
Constant	A named location in memory storing a single piece of data that cannot change.
Array	A named location in memory that can hold multiple pieces of data of the same type.
SQL	A language used to retrieve and manipulate data in a database.
Sub programs	A named section of code which completes a sub task that can be reused.
Function	A type of sub program that returns a value.
Procedure	A type of sub program that doesn't return a value.
Comparison operator	An operator used to compare two values. Commonly used in an if statement.
Arithmetic operator	An operator used to carry out a mathematical function such as addition or subtraction.
Casting	Converting one data type to another
Concatenation	Joining two or more stings together

Data Types

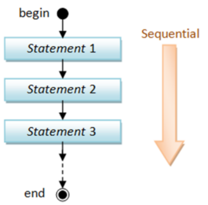
Integer	A whole number, e.g. -1, 3 etc.
Float/ Real	A decimal number, e.g. 1.4
Boolean	A true or false value
Char	A single character, e.g. a
String	A combination of characters, e.g. 'hello'

Programming Constructs

Sequence

A set of instructions in order.

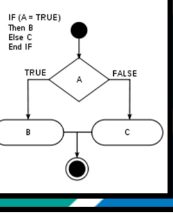
```
OUTPUT "1"
OUTPUT "2"
OUTPUT "3"
OUTPUT "4"
OUTPUT "5"
```

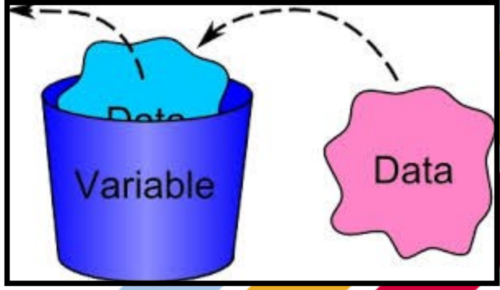


Selection

Where the algorithm makes a decision based on a choice of different paths.

```
weather ← USERINPUT
IF weather = "Rain" THEN
        OUTPUT "Take a brolly"
ELSE
        OUTPUT "Have a nice day"
```



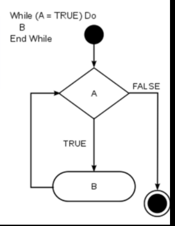


Iteration

Also known as a loop, this is the process of repeating a set of instructions.

```
FOR i ← 1 TO 5
        OUTPUT i
```

```
a←1
WHILE a < 6 THEN
        OUTPUT a
        a←a+1
```



Mildenhall College
ACADEMY

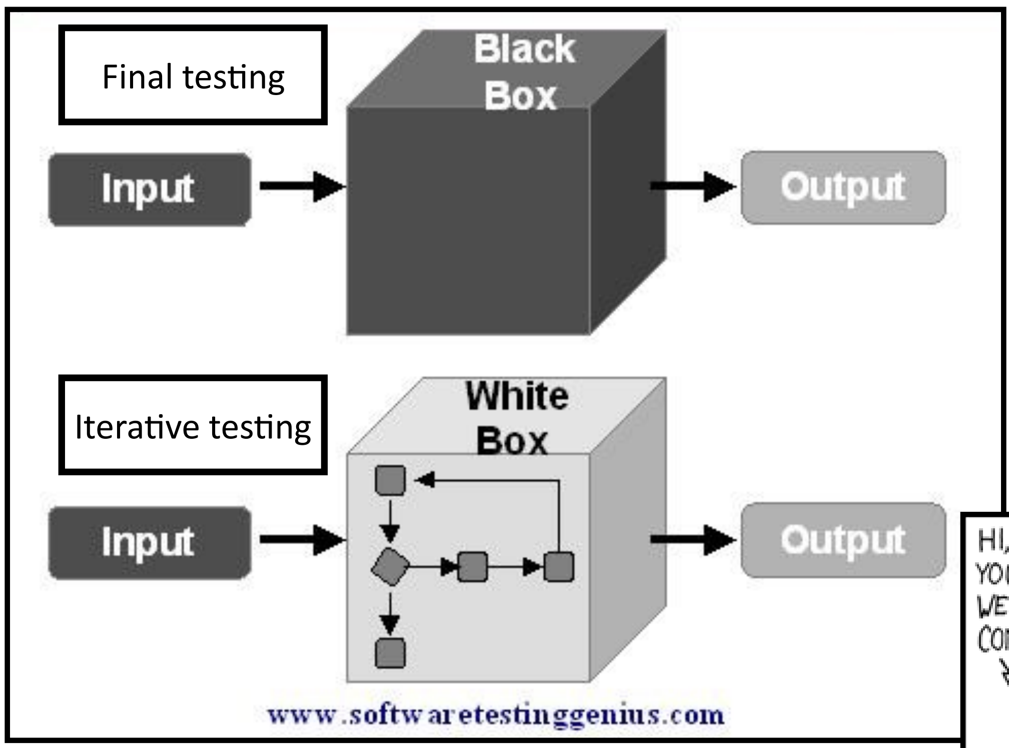
2.3 Producing Robust Programs Knowledge Organiser

Key learning

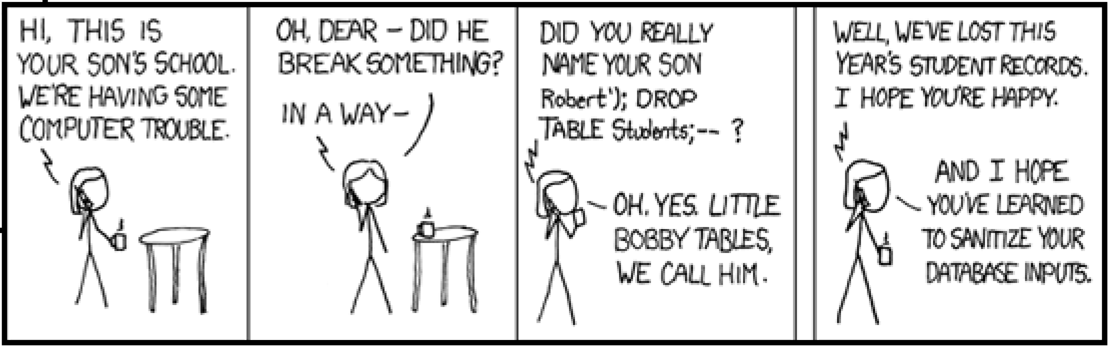
- **Defensive design considerations:**
  - **Anticipating misuse**
  - **Authentication**
- **Input sanitisation/validation**
- **Maintainability:**
  - **Comments**
  - **Indentation**
  - **Use of functions**
  - **Sensible variable names**
- **The purpose of testing**
- **Types of testing:**
  - **Iterative**
  - **Final/terminal**
- **How to identify syntax and logic errors**
- **Selecting and using suitable test data**
- **Refining algorithms**

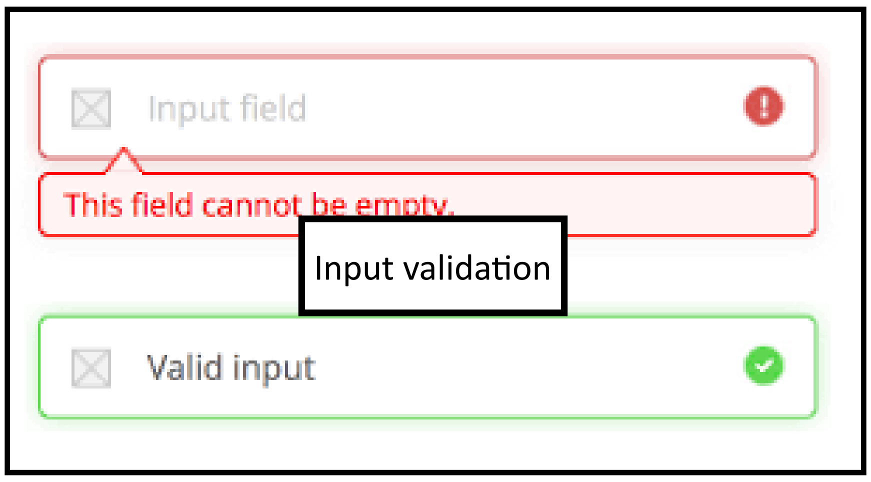
Key terms

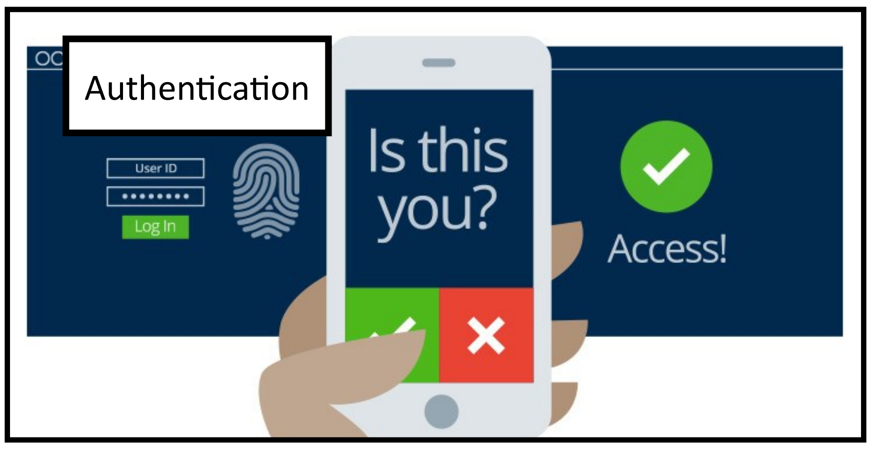
Input sanitation	Removing unwanted characters, such as spaces or punctuation, from inputs.
Input validation	Checking that an input is reasonable, for example age needs to be > 0.
Contingencies	Planning for when something doesn't work as expected.
Authentication	Making sure user have to sign in to access and change data.
Normal test data	Data that should be accepted.
Boundary test data	Data that should be accepted but is borderline.
Erroneous test data	Data that should not be accepted.
Syntax error	An error that causes a program to stop running due to the code not following the rules of the language.
Logic error	An error where the program can still run but will not give the expected output.
Runtime error	An error where the program will stop running due to the program not being able to carry out the instruction.

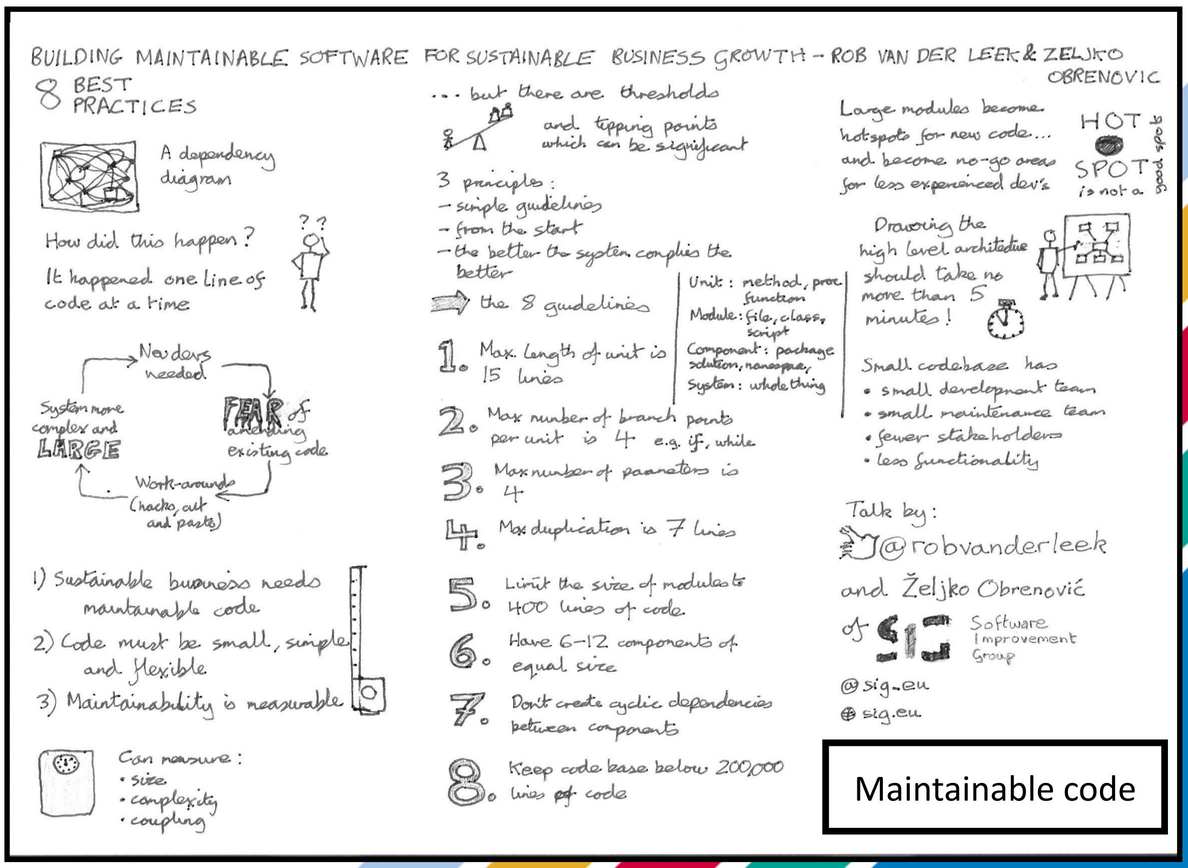

Final testing

Iterative testing


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.


Input validation


Authentication


BUILDING MAINTAINABLE SOFTWARE FOR SUSTAINABLE BUSINESS GROWTH – ROB VAN DER LEEK & ZELJKO OBRENOVIC

8 BEST PRACTICES

A dependency diagram

How did this happen?

It happened one line of code at a time

New devs needed → FEAR of amending existing code → Work-arounds (hacks, cut and paste) → System more complex and LARGE

1) Sustainable business needs maintainable code

2) Code must be small, simple and flexible

3) Maintainability is measurable

Can measure:
- size
- complexity
- coupling

... but there are thresholds and tipping points which can be significant

3 principles:
- simple guidelines
- from the start
- the better the system complies the better

the 8 guidelines

Unit: method, proc. function
Module: file, class, script
Component: package solution, namespace,
System: whole thing

1. Max. Length of unit is 15 lines
2. Max number of branch points per unit is 4 e.g. if, while
3. Max number of parameters is 4
4. Max duplication is 7 lines
5. Limit the size of modules to 400 lines of code
6. Have 6–12 components of equal size
7. Don't create cyclic dependencies between components
8. Keep code base below 200,000 lines ~~of~~ code

Large modules become hotspots for new code... and become no-go areas for less experienced dev's

HOT SPOT is not a good spot

Drawing the high level architecture should take no more than 5 minutes!

Small codebase has
- small development team
- small maintenance team
- fewer stakeholders
- less functionality

Talk by:
@robvanderleek
and Željko Obrenović
of SIG Software Improvement Group
@sig_eu
sig.eu

Maintainable code

Mildenhall College
ACADEMY

2.4 Boolean Logic Knowledge Organiser

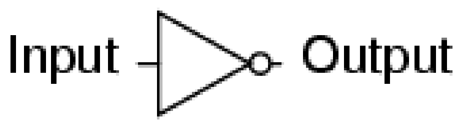
Key learning

- Why data is represented in computer systems in binary form
- Simple logic diagrams using the operations AND, OR and NOT
- Truth tables
- Combining Boolean operators using AND, OR and NOT to two levels
- Boolean notation
- Applying logical operators in appropriate truth tables to solve problems

Key terms

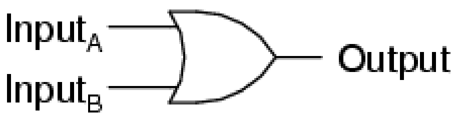
Binary	A series of 1s and 0s used for data and instructions represented by switches/ transistors.
Boolean logic	A form of logic centred around operations between combinations of 1s ad 0s.
AND (Conjunction)	A Boolean operation where both inputs must be a 1 for the output to be 1.
OR (Disjunction)	A Boolean operation where at least one input needs to be a 1 for the output to be 1.
NOT (Negation)	A Boolean operation where the output is the inverse of the input.
Truth table	A table which can be used to work out the output for different combinations of inputs being used with Boolean operators.
Logic diagram	A way to visualise how data passes through different gates.

NOT gate truth table



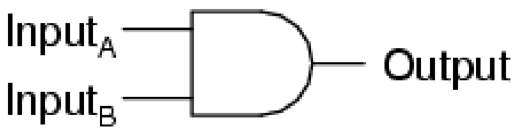
Input	Output
0	1
1	0

2-input OR gate

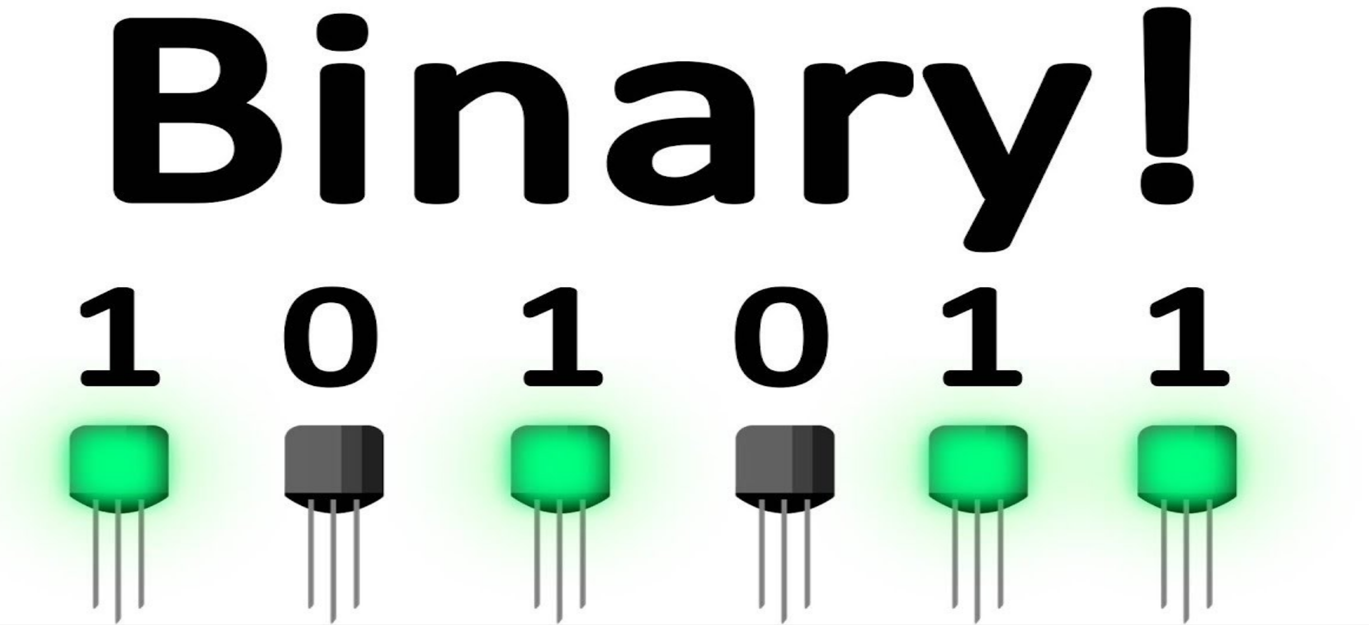


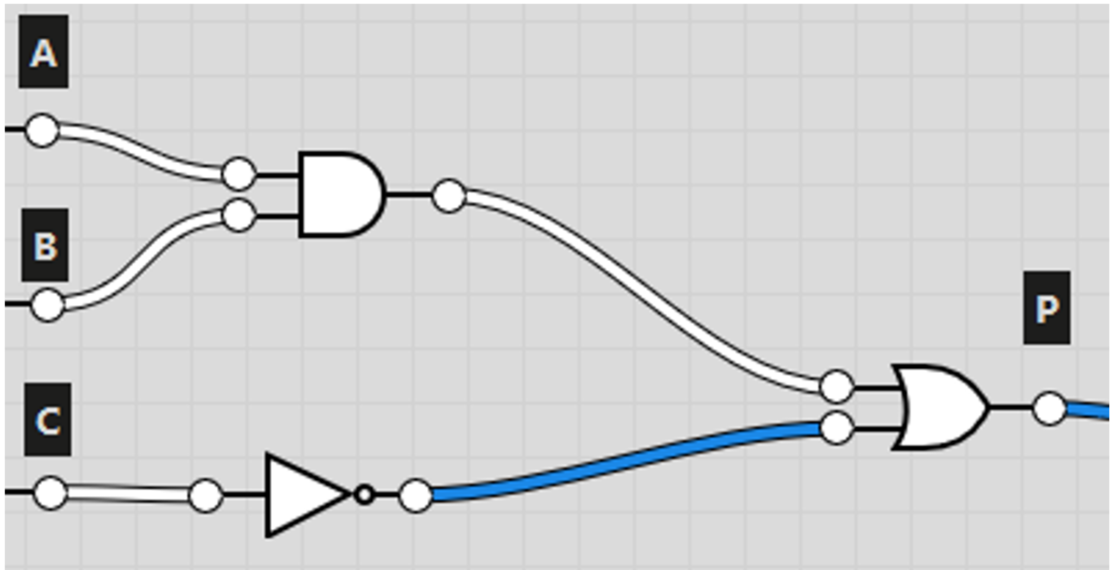
A	B	Output
0	0	0
0	1	1
1	0	1
1	1	1

2-input AND gate



A	B	Output
0	0	0
0	1	0
1	0	0
1	1	1





Creating logical expressions

- P = (A AND B) OR NOT C
- $P = (A \land B) \lor \neg C$

Mildenhall College
ACADEMY

2.5 Programming languages and IDEs Knowledge Organiser

Key learning

- **Characteristics and purpose of different levels of programming language, including low level languages**
- **The purpose of translators**
- **The characteristics of a compiler and an interpreter**
- **Common tools and facilities available in an integrated development environment (IDE):**
    - **Editors**
    - **Error diagnostics**
    - **Run-time environment**
    - **Translators**

Key terms

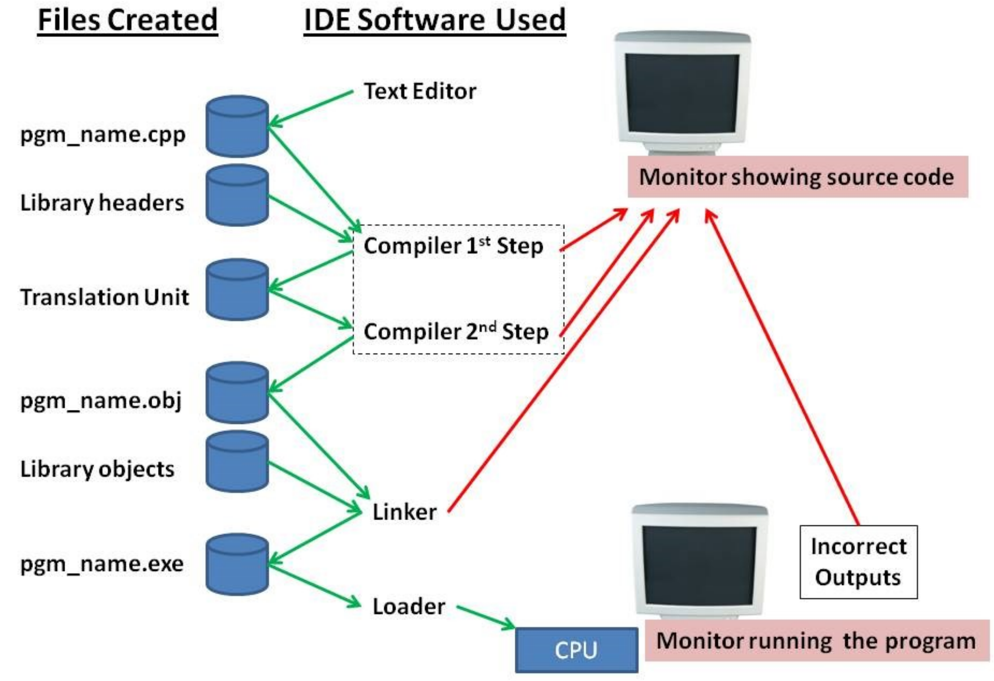
High level language	A programming language which closely resembles English, for example Python.
Low level language	A programming language in Binary designed for the CPU to understand.
Compiler	A translator that converts high level code into low level code in one go.
Interpreter	A translator that converts high level code to low level code line by line.
Editor	A program designed to make writing code easier usually including a range of tools such as colour coding.
Run-time environment	A program which executes code written and allows it to be checked for errors.
IDE	**Integrated Development Environment** is software that normally combines editors, debuggers, and translators.

Code editor tools

- Auto indent
- Colour coding/ Syntax highlighting
- Auto complete code
- Code suggestion
- Find and replace
- Live preview

Compiler	Interpreter
Takes entire program as input	Takes a single instruction at a time as input
Creates an intermediate object code	Doesn't generate object code
Code is compiled before being executed	Translation and execution take place at the same time
Faster to run once compiled	Slower to run
Displays all errors at the end of compilation	Displays each error as it finds it
Error detection is more difficult	Error detection is easier

High level language	Low level language
Close to English	Close to the native language of the computer
Faster to write	Written in Binary
Can run on any machine as long as suitable translator is used	Can only run on one device
No knowledge of hardware is needed	Linked to specific hardware
Examples include Python, C++, Java and Visual Basic	Also known as machine code


Files Created
IDE Software Used
Text Editor
pgm_name.cpp
Library headers
Compiler 1st Step
Translation Unit
Compiler 2nd Step
pgm_name.obj
Library objects
Linker
pgm_name.exe
Loader
CPU
Monitor showing source code
Incorrect Outputs
Monitor running the program