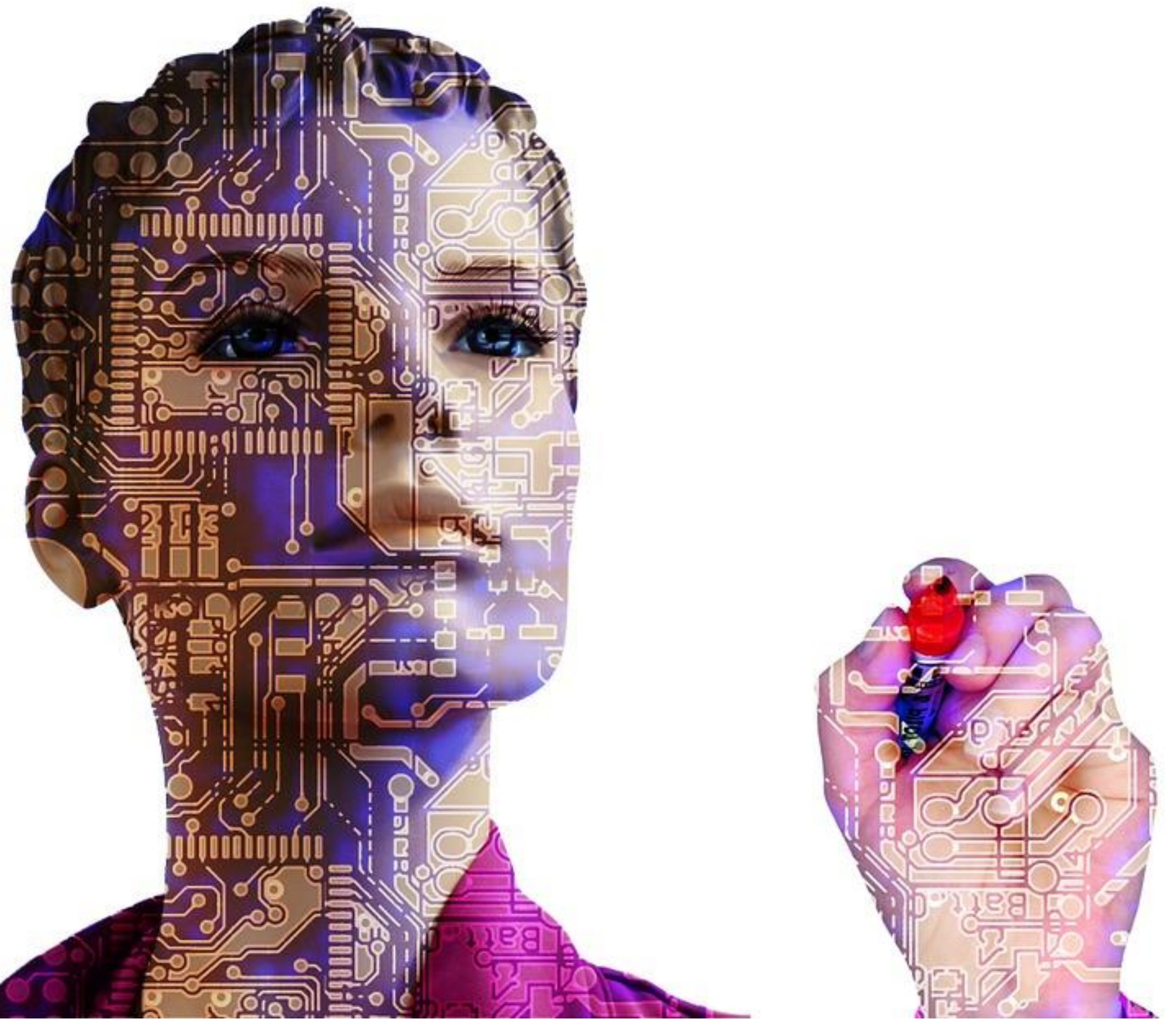
GCSE Computing Revision Guide

OCR Specification

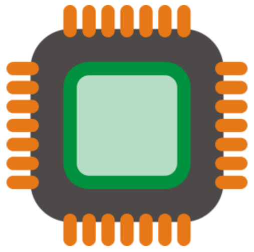


Contents

Systems Architecture
CPU

- Purpose of the CPU
- How common characteristics of CPUs affect their performance

Specification — Revised



ⓘ The purpose of the CPU is to carry out the processing of data on the computer system

ⓘ It performs the fetch-decode-execute cycle.

ⓘ The CPU fetches, decodes and executes instructions

The Performance of the CPU

Clock speed	Cache Size	Number of Cores
This determines the rate at which instructions are carried out each second The clock speed is measured in Hertz (Hz) A 3.6 Ghz processor carried out 3.6 billion calculations a second	Cache memory is a buffer that sits between the CPU and main memory. The CPU will check here first for instructions that have been fetched before The larger the cache the more space there is for instructions the CPU needs The cache has similar access speeds to the CPU and is therefore quicker to fetch instructions from	A core is an independent processor in the CPU A dual core has 2, quad core 4, hex core 6 processors working simultaneously The higher the number of cores the better performance of the computer as it can multitask

- Clock speed – number of fetch-decode-execute cycles a second
- Cache Size – high speed memory used by the CPU
- No of Cores – number of independent processors in the CPU working together

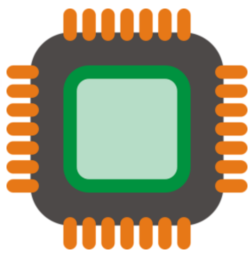
ⓘ The CPU contains **registers** which are temporary memory stores within the Cpu which have a specific purpose.

Systems Architecture

Von Neumann Architecture

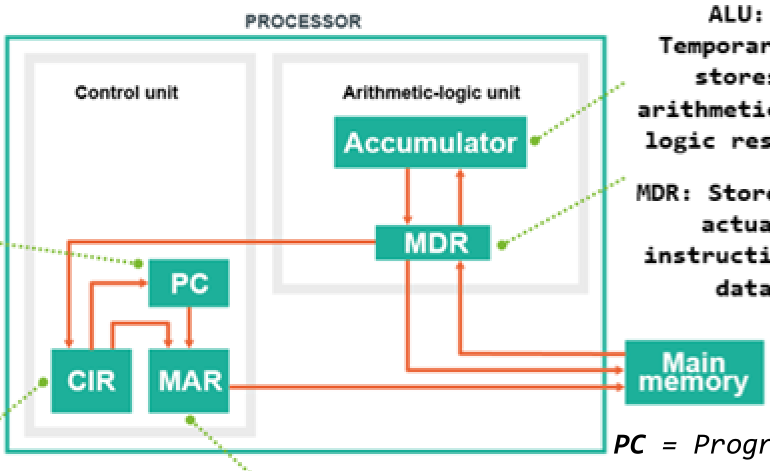
Von Neumann Architecture and the purpose of different registers

Specification — Revised



ⓘ The Von Neumann Architecture describes a system where data and programs/instructions are stored in the same main memory location

ⓘ The fetch-decode-execute cycle is the process of fetching these instructions from memory and executing them


PROCESSOR
Control unit
Arithmetic-logic unit
Accumulator
MDR
PC
CIR
MAR
Main memory
ALU: Temporarily stores arithmetic and logic results
MDR: Stores the actual instruction or data
PC: Points to the next instruction
CIR (Current Instruction Register): Holds the current instruction to be executed
MAR: Stores the address of the instruction/data

PC = Program Counter

MAR = Memory Address Register

MDR = Memory Data Register

CIR = Current Instruction Register

The steps in the cycle:

Fetch	Decode	Execute
ⓘ Address from the PC is copied to the MAR ⓘ Instruction from MAR is fetched and copied to MDR ⓘ The instruction in the MDR is copied to the CIR ⓘ Increment the PC	ⓘ The instruction in CIR is decoded by the control unit ⓘ Data may be loaded into the MDR	ⓘ The instruction is performed ⓘ The ALU may be used for any logic or calculations ⓘ The result is stored in the accumulator

Systems Architecture

Embedded Systems

- Purpose of embedded systems
- Examples of embedded systems

Specification

Revised

ⓘ An embedded system is one which has a processor built in to another device

ⓘ A computer System that is made up of both Hardware and Software often known as Firmware

ⓘ Usually for very specialised tasks

ⓘ Doesn’t usually contain an Operating System

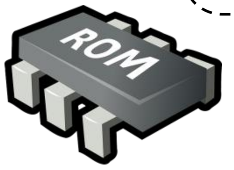
Examples:
- Dishwasher
- Microwave
- Fridge
- Smart phone
- TV

Memory
RAM and ROM

- The difference between RAM and ROM
- The purpose of RAM in a computer system
- The purpose of ROM in a computer system
- The need for Virtual Memory
- Flash Memory

Specification

Revised



RAM and ROM

Random Access Memory (RAM)	Read Only Memory (ROM)
Purpose = Stores data and programs currently being used by the computer	**Purpose** = Stores instructions needed to start up the computer – contains the boot program
ⓘ Can be changed by the computer at any time	ⓘ Programmed during the computers manufacture and cannot normally be changed
ⓘ Volatile memory (data is lost when the power is turned off)	ⓘ Non-Volatile (data is not lost when the power is turned off)
ⓘ Larger memory (Starting at 4GB in most computers)	ⓘ Small (Only MB needed for the boot program)
ⓘ The more RAM the more programs that can be run at the same time. Allows for more multitasking.	ⓘ ROM is needed as it is always there to start the computer

Flash and Virtual Memory

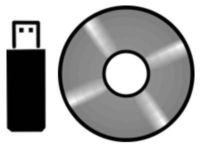
Flash	Virtual
ⓘ Flash memory is type of non-volatile (ROM) memory that can be changed and does not need a power supply to keep its contents ⓘ There are no moving parts which make it fast and reliable ⓘ Examples of flash memory in use: ⓘ Memory cards in digital cameras. ⓘ Mini/Micro SD cards in Smartphones. ⓘ USB memory sticks. ⓘ Solid state drives	ⓘ Virtual memory is part of the hard drive used as an extension to RAM. If there is not enough RAM to hold all the data and run the programs needed then it will make use of some of the hard drive. ⓘ Access speeds from the hard drive are slower than from RAM. More RAM reduces the need for virtual memory which improves performance.

Storage

Secondary Storage

- The need for secondary storage
- Data capacity and calculation of data capacity requirements
- Common types of storage: Optical, Magnetic, Solid State

Specification — Revised



ⓘ Secondary storage is needed to store our files and programs when they are not in use. These commonly fall into three categories: Optical, Magnetic and Solid State.

ⓘ Secondary storage is long term, non-volatile storage

Optical	Magnetic	Solid State/Flash
ⓘ Lasers write data to the surface of a disk. ⓘ Optical media includes: CD, DVD, Blu-Ray ⓘ Excellent for distributing software ⓘ Good capacity ⓘ Low cost ⓘ Light and portable ⓘ Can get damaged over time ⓘ Slow access speed	ⓘ The magnetic tape is moved along a read-write head inside a disk drive ⓘ Examples: hard disk and tapes ⓘ Used for backups ⓘ High capacity ⓘ Cheap ⓘ Reliable ⓘ Slow to read due to moving parts	ⓘ No moving parts make solid state memory have a very fast access speeds. Most are a type of flash memory ⓘ Examples: USB drives, memory cards, solid state hard drives ⓘ Large capacity but less than magnetic tape ⓘ More expensive ⓘ Portable ⓘ Reliable and not affected by being moved around

Calculating Storage Requirements

ⓘ From the section on data representation we know how many bits are in a byte and how many bytes in a kilobyte etc. We also looked at how to calculate the size of an image.

ⓘ Using the knowledge we can calculate how much storage will be needed in different scenarios. For example:

A text file that contains 10000 characters. Give your answer in KB

We know that each character is 1 byte in ASCII. So 10000 x 1 = 10000. There are 1024 bytes in a kilobyte so 10000/1024 = 9.77kb

Storage

Secondary Storage

Suitable storage devices and storage media for a given application, and the advantages and disadvantages of these, using characteristics:

- Capacity
- speed
- portability
- durability
- reliability
- cost

Specification Revised

When choosing an appropriate device for a given scenario you need to consider the following characteristics:

- **Capacity:** How much space there is to store files.
- **Access Speed:** How quickly the computer can read and write data to or from a storage device or write data to it.
- **Portability:** Can you easily unplug it and carry it away?
- **Durability:** How easily is it damaged? Will it survive being dropped?
- **Reliability:** Can the data always be accessed and be correct?
- **Cost:** How expensive is the storage device

Quick Comparison:

Access Speed

Slowest *Optical Disk* *Magnetic Tape* *HDD* *USB/Flash* *SDD* Fastest

Cost

Cheap *Magnetic Tape* *Optical Disk* *HDD* *USB/Flash* *SDD* Expensive

Capacity

Small *Optical Disk* *USB/Flash* *SDD* *HDD* *Magnetic Tape* Large

Portability

Difficult *SDD* *HDD* *Magnetic Tape* *Optical Disk* *USB/Flash* Easy

Durability/Reliability

Low *Optical Disk* *Magnetic Tape* *USB/Flash* *SDD* *HDD* High

Wired and Wireless Networks

Types of Networks

- Types of Networks: LAN and WAN
- Hardware needed to connect stand-alone computers into a Local Area Network

Specification — Revised

ⓘ A network is 2 or more computers connected to each other.

Advantages of a network:
1. It allows communications between workers.
2. It allows data to be shared.
3. It allows peripherals to be shared.
4. It allows computers to be upgraded more easily.
5. It allows distributed processing

WAN	LAN
ⓘ Wide Area Network ⓘ Computers are geographically remote/long distance away ⓘ Communication medium is not owned by the company	ⓘ Local Area Network ⓘ Covers small geographic area located on one site ⓘ All hardware for a LAN is owned by the organisation that uses it

Hardware Needed

Network Interface Card (NIC)	Wireless Access Points
ⓘ Any device connected to a network needs to have a NIC (normally built into the motherboard) ⓘ Produces electrical signals for receiving and sending messages on the network ⓘ Has a unique MAC address	ⓘ Allows wireless devices to connect to a network ⓘ A type of switch for wireless devices
Router/Switch	**Transmission Media**
ⓘ A switch connects devices together on a LAN. ⓘ They send data along a network using the MAC address for the destination. ⓘ A switch sends data to the intended destination only ⓘ A router is used to send data between networks and connect devices to the internet	ⓘ Physical wires that connect devices together in a LAN. ⓘ Twisted Pair (Cat5e/Cat6): Most common. Cheap and easy to install. Fast and reliable transmission. ⓘ Coaxial: Bulkier than twisted pair. Made of a single copper wire which is insulated to minimise interference ⓘ Fibre Optic: Can transmit data as light. High performance and do not suffer from interference. Can transmit over large distances but is expensive

Wired and Wireless Networks

Client Server and Peer to Peer Networks

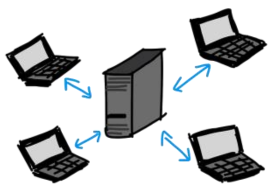
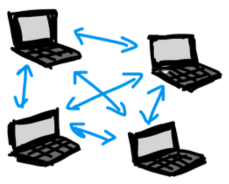
- The different roles of computers in a client server and a peer to peer network
- Factors that affect the performance of networks

Specification

Revised

ⓘ There are two main ways that networks can be organised. Once involves a server and the other does not

Server = A server is essentially a more powerful computer that manages a network and provides services to the clients. Some of the functions of the server are: to share files, provide security, provide access to programs and to backup files.

Client Server	Peer to Peer Network
 ⓘ Has a designated server. The nodes are clients and make requests to access files or programs from the server. ⓘ The server is a powerful machine typically with more memory that can serve the needs of the clients. ⓘ Handles the security (logins), backup and installation of software ⓘ Expensive to set up and you need to have a specialist to maintain the network ⓘ If the server goes down the clients cant access the network	 ⓘ All computers are equal ⓘ No server ⓘ As there is no server backups and updated need to be done individually ⓘ Copying of files between devices creates duplicates ⓘ Tend to be slow ⓘ Security is up to each user ⓘ Suitable when a small network such as a home network

Performance of Networks

- ⓘ Bandwidth = the amount of data that can be transferred in a given time. The higher the bandwidth the better the performance
- ⓘ Too many users or heavy use can slow down the network
- ⓘ Wired connections are generally faster than wireless connections and more secure
- ⓘ Hardware used to setup network can have an effect
- ⓘ Network topology used

Wired and Wireless Networks

The Internet

- The internet as a worldwide collection of computer networks:
  - DNS (Domain Name Server)
  - Hosting
  - The Cloud
- The concept of virtual networks

Specification — Revised

The Internet = A collection of worldwide networks. It is the largest WAN connecting networks all around the world.

World Wide Web = is not the same as the Internet. The WWW is a collection of websites

Hosting = A host is another computer that stores a particular resource. For example websites are hosted on other computers/servers which we access by typing in the address.

The Cloud	
Uses the Internet to store files and applications that we can access remotely. An example is Google Drive where we can access out files from anywhere with an Internet connection.	
Advantages	**Disadvantages**
ⓘ Can offer increased storage when needed ⓘ Easy to share files ⓘ Can access work and files from anywhere as long as you are connected ⓘ Provides security and backup automatically for you ⓘ Don't need to pay IT staff to manage the hardware and there is no need to purchase expensive hardware	ⓘ Need a connection to the internet ⓘ Need to trust someone else to backup and secure your data ⓘ Can be vulnerable to hackers ⓘ Subscription fees may be expensive ⓘ There are issues with who owns the data stored on the cloud

Wired and Wireless Networks

The Internet

- The internet as a worldwide collection of computer networks:
  - DNS (Domain Name Server)
  - Hosting
  - The Cloud
- The concept of virtual networks

Specification — Revised

Domain Name System – DNS

Websites all have a unique IP address which is used to access them. These are hard to remember so they have a domain name which is used in the uniform resource locator (URL) such as www.google.co.uk.

The DNS translates these URLs or domain names into the IP address so you don't need to remember them.

Benefits of using a DNS

- Constantly updated by other DNS servers
- When you request an address(URL), the DNS server looks up the URL and returns the IP address, or
- Searches for the address from other DNS servers
- People do not need to remember IP addresses
- As long as you are connected to a DNS server you can have access to all the addresses

Virtual Network

Virtual Network = A network that is software based. It uses the same existing physical network but creates other individual networks without having to rewire them.

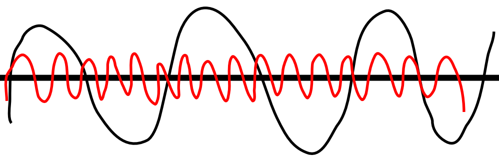
- Each virtual network has its own security and firewall
- Several virtual networks can exist on the same physical network
- Virtual networks can also be created by using the services of the cloud
- An example could be in a school where a virtual network is set up which allows all the students to be connected and this is separate to the virtual network for all the admin staff

Network Topologies Protocols and Layers

Wi-Fi

- Frequency and Channels
- Encryption

Specification



Revised

To connect to a wireless network the following hardware devices can be used: Wireless Access Point, Router, Modem, Hub.

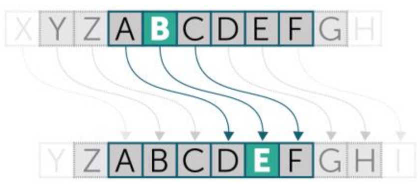
Frequency	Channels
ⓘ Frequency is the rate at which the signal changes ⓘ The number of times it repeats per unit of time (GHz) ⓘ Wifi uses two radio frequency bands. 2.4 GHz and 5 GHz.	ⓘ A channel is the range of frequencies that will transmit data ⓘ Channels overlap ⓘ 2 devices using the same channel may suffer from interference ⓘ Changing the channel can reduce interference.

Encryption

Data that is transmitted over a network can be intercepted. Encryption is used to prevent the data being understood if it is intercepted.

Encryption = putting the data into a code that cannot be understood unless you have the key to decrypt it.

The most common encryption security for Wi-Fi is WEP and WPA

Caeser Cipher	Symmetric Encryption	Public Key (Asymmetric Encryption)
Letters are shifted by a given number 	Same key used to encrypt and decrypt a message	Two keys! A public key known to everyone for encrypting and a private, secret key for decrypting.

Network Topologies Protocols and Layers

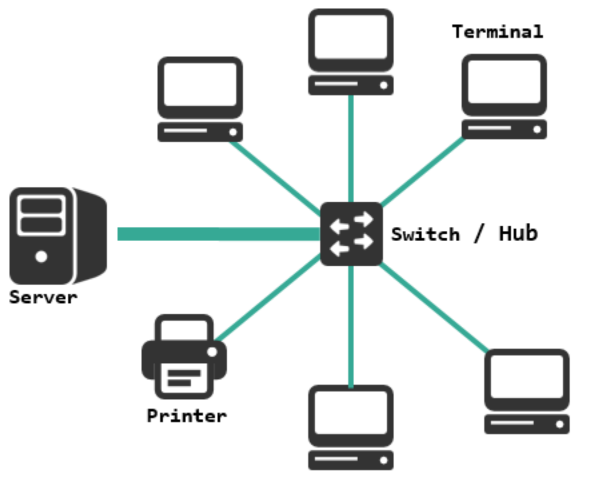
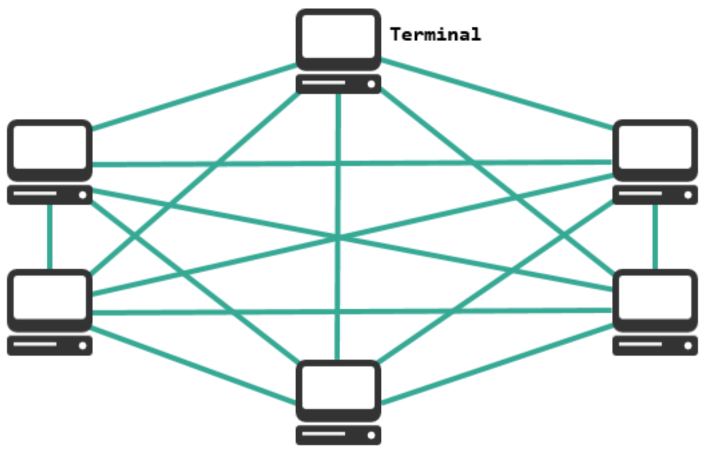
Star and Mesh Topologies Packet Switching

- Star and Mesh Network Topologies
- Packet Switching
- Ethernet

Specification

Revised

Topology = the layout of the network

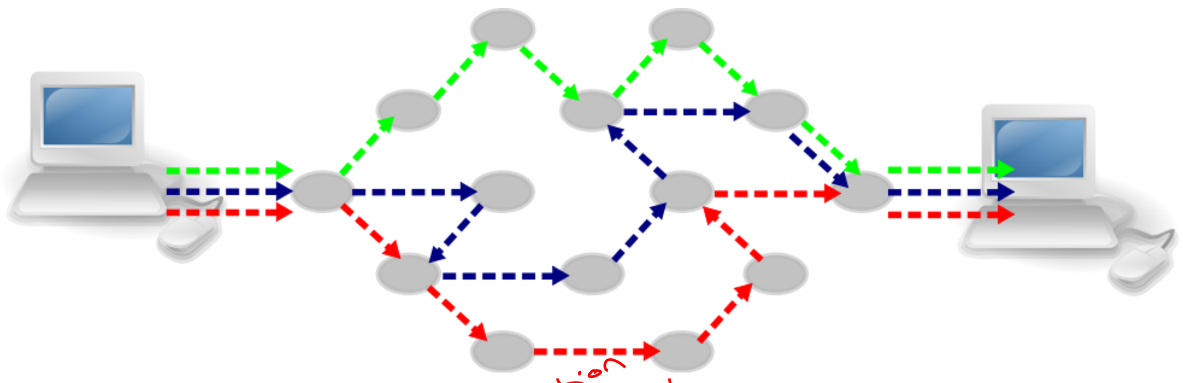
Star Network	Mesh Network Network
ⓘ All computers are connected to a central switch or server that controls the network ⓘ Data is sent to the server which then sends it to another device.	ⓘ A decentralised network that allows devices to be connected directly or indirectly to each other ⓘ Data is sent along the fastest route from one device to another ⓘ You can have a full or partial mesh
	
ⓘ Fast data transfer as there are fewer collisions ⓘ If one cable fails the other terminals are not affected ⓘ If the central device (switch or server) goes down the whole network goes down	ⓘ Data can be trasmitted to different devices simultaneously ⓘ An involve redundant connections ⓘ Network maintenance and administration can be difficult

Packet Switching

= Splitting data to be sent over a network into equal sized packets and then they are sent tacking different routes

- ⓘ The computer splits the file into packets
- ⓘ Each packet is of a fixed size
- ⓘ The packets are given a header including the destination address and the packet number
- ⓘ Packets find their own way across the network to the destination
- ⓘ Server waits until all packets have arrived
- ⓘ Server reorders packets to create the file
- ⓘ Any missing / non-arriving packets are re-requested
- ⓘ error checking is performed on receipt of packets

Contents of a Packet		
Header	**Payload**	**Footer**
ⓘ Destination Address ⓘ Packet number	ⓘ The data that is being sent	ⓘ Error checking such as a check sum



Ethernet

Ethernet is a protocol

Transport connection protocol

internet protocol

- ⓘ It is within the TCP/IP stack
- ⓘ It governs the connection of devices over the internet
- ⓘ Governs the transmission of data between devices over the internet
- ⓘ Uses cables to transmit data between devices in a LAN Local area network

Protocol = *Rules/Agreed ways of doing something such as connecting or communicating*

Network Topologies Protocols and Layers

Protocols and Layers

- Protocols that include TCP/IP, HTTP, HTTPS, FTP, POP, IMA, SMTP
- The concept of Layers

Specification — Revised

Protocol = *Rules/Agreed ways of doing something such as connecting or communicating*

TCP/IP = *Protocol for how data is sent between networks*

TCP= Transmission Control Protocol	IP = Internet Protocol
ⓘ Sets the rules for how devices connect to a network ⓘ Splits the data into packets ⓘ Reassembles the packets at the other end ⓘ Checks data is correctly sent and delivered	ⓘ Is responsible for the packet switching WAN

Other Protocols

HTTP = Hyper Text Transfer Protocol	Used by websites and to communicate with web servers
HTTPS = HTTP Secure	A secure version of HTTP
FTP = File Transfer Protocol	Uses to send or retrieve files to or from a server LAN
POP3 = Post office Protocol 3	Retrieve emails from a server. Held until downloaded and then deleted
IMAP = Internet Message Access Protocol	Retrieve emails from a server. Stored on server until deleted. Can view from several devices.
SMTP =Simple Mail Transfer Protocol	Used to send emails between servers

Layers = *Group of protocols that share similar functions*

A
T
N
D

Layer Name	Protocols in Layer	Examples
4. Application Apps	Selects the correct protocol depending the application. E.g. sending an email or viewing a website	*HTTP, FTP, SMTP*
3. Transport	Controls the data flow and splitting data into packets	*TCP*
2. Network	Making connections and controlling the packet switching directing the data packets	*IP*
1. Data Link	Physical hardware that connects 2 hosts such as the NIC and cabling	*Ethernet*

Data can be passes between adjacent layers. E.g. layer 2 can pass data to layer 1 and 3, but layer 1 can only pass data to layer 2

- ⓘ Self-contained which allows them to function without affecting the other layers
- ⓘ They can be changed without affecting the other layers
- ⓘ *It allows network communication to be broken down into manageable pieces*
- ⓘ *Having set rules(protocols) ensures that companies make compatible hardware and software*

Systems Security

Forms of Attack and Threats

- 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
- Poor Network Policy

Specification | Revised

Virus	Software that replicates itself Deletes data // fills hard drive space // slows computer
Malware	Malicious software that can take different forms such as Viruses(program that replicates itself and causes damage), Trojan horses(software that tricks the user into installing it), spyware (gathers information about the user)
Phishing	Using fraudulent emails to try and obtain your personal information such as passwords and credit card numbers. Often present to be from reputable companies
Brute Force	A method of trying to gain access to data by tying all possible combinations to discover a user's password.
Denial of Service	Designed to shut down a network or webserver by flooding it with traffic that it cannot handle.
Social Engineering	Social engineering is the art of manipulating people so they give up confidential information
SQL Injection	SQL is the language used to control databases. Malicious code entered into a website form to modify he SQL statement that is executed resulting in either: unauthorised access to data; modification of data; deletion of data; insertion of data Can be prevented by validating user input

Systems Security

Identifying and Preventing Vulnerabilities

- Penetration testing
- Network forensics
- Network Policies
- Anti-Malware software
- Firewalls
- User access levels
- Passwords
- Encryption

Specification

Revised

Network Policy	ⓘ Agreed procedures for people in an organization to follow. This sets out what you can and can't do. For example not being allowed to use external storage devices or using email for personal use.
Penetration Testing	ⓘ Testing by simulating a possible attack in order to identify any vulnerabilities
Network Forensics	ⓘ Monitoring and analysis of computer network traffic to check for intrusion
Anti-Malware Software	ⓘ Software that is designed to detect and remove malware from infecting the computer.
Firewalls	ⓘ Prevents unauthorised access to a network by checking incoming traffic.
User Access Levels	ⓘ Limiting the access to information depending on the type of user. For example an administrator would have access to all files whereas an employee at a lower level would only see he files they need.
Strong Password Features	ⓘ A strong password is one that is difficult to guess. A strong password will be at least 6 characters and use upper and lower cases, numbers and symbols.
Encryption	ⓘ Encoding data so that it cannot be understood if intercepted without the key.

Systems Software
Operating System

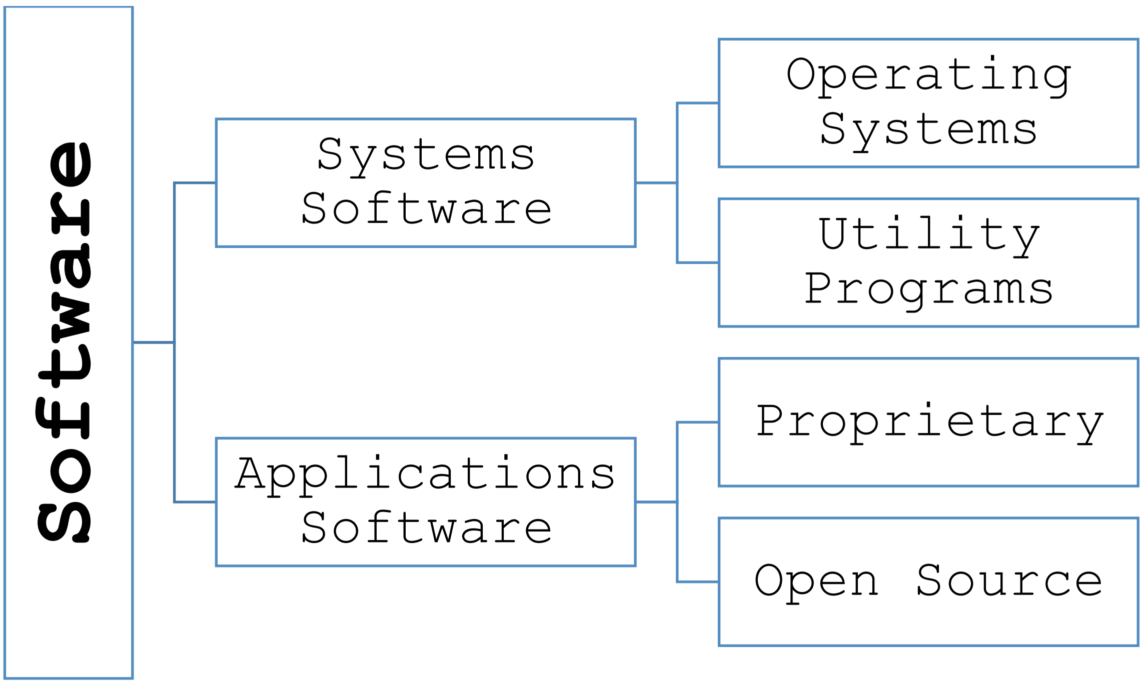
- The purpose and functionality of systems software
- Operating systems:
  - User Interface
  - Memory Management/Multitasking
  - Peripheral Management and Drivers
  - User Management
  - File Management

Specification — Revised

Systems Software = Software designed to maintain and control the hardware, it also provides and interface between software and hardware. The two main types of systems software are operating systems and utility programs

Operating system = controls the hardware and software/provides and interface for the user

Utility programs = help to maintain the computer



Function	Description

User Interface	ⓘ Allows the user to communicate with the hardware.
	Different Types: Command line, Graphical User Interface (GUI), Menu Drive, Voice
	For a GUI remember WIMP for the key features:
	W = Windows
	I = Icons
	M = Menus
	P = Pointers

Memory Management/Multitasking	ⓘ Allocates memory to applications – can run more than one piece of software at a time ⓘ Removes data no longer needed – frees up space for other programs ⓘ Moves data between **RAM and Virtual Memory** – allows more programs to run ⓘ **Multitasking:** The OS can run multiple applications at the same time ⓘ They are taking it in turns to get processor time to execute instructions ⓘ The OS must manage how the processes share the processor
Peripheral Management and Drivers	ⓘ A **device driver** is a program that controls a peripheral device such as a printer. It allows the operating system to communicate with the device ⓘ A **peripheral** is an external device not directly connected to the CPU such as a monitor or keyboard ⓘ The operating system deals with taking input from the devices and sending output to them.
User Management	ⓘ Provides user **names and passwords** for security ⓘ **Levels of access**. Gives different users access to specific data or resources ⓘ Operating systems can be single user or multi user. A multiuser OS allows several users to use the computer at the same time (e.g. for a mainframe or supercomputer).
File Management	ⓘ Stores data in a hierarchical structure which allows data to be stored in a organised way ⓘ Manages where data is stored and the user does not need to know how ⓘ Allows files and data to be retrieved and edited.

Systems Software

Utility Systems Software

- Utility Systems software:
  - Encryption Software
  - Defragmentation
  - Data Compression
  - The role and methods of backup:
  - Full
  - Incremental

Specification

Revised

Encryption	ⓘ Puts data in a code (encrypts) that cannot be understood unless the user has a key to decrypt it. ⓘ Prevents unauthorised people accessing the data if it is intercepted.
Defragmentation	ⓘ Data when stored on a hard disk can get fragmented (split up) which makes it slower to access the data. Data is saved in the available spaces and not always grouped together. ⓘ This can happen as files are moved, deleted or changed size. ⓘ Defragmentation reorganises the data so that it is stored together
Compression	ⓘ Reduce the size of files before sending or saving ⓘ So they can be transmitted faster or take up less space ⓘ Allows media on websites to load faster
Backup	ⓘ A backup is making a copy of the original file ⓘ A full backup copies every files on the system. This can can take time and a lot of space ⓘ An incremetal backup only copies files that have changed since the last backup was made. This is faster to create and uses less space. If a full restore is needed this is slower than resroring from a full backup

Ethical, Legal, Cultural and Environmental Concerns

Open Source vs Proprietary Software

Open Source vs Proprietary Software

Specification — Revised

Proprietary

Proprietary software cannot be copied/altered (without permission of the copyright owner). It is distributed only as a compiled program/source code not available

Advantages	Disadvantages
Support is available if there is a problem Updates are regularly available Will have tested extensively	Can be expensive as you have to purchase licenses It may be inflexible to the user's needs Can tie you in to uses one particular platform

Open Source

Open source software is distributed with its source code and can be modified

Advantages	Disadvantages
License free and its source code is available to others to use and improve It can be altered as the source code is available Can be reliable as many people modify and improve it	May not be anyone to contact if something goes wrong Updates may not be available or be irregular

Why you would choose to make your software Open Source:

- Other people can improve and maintain your code
- You can receive an income from advertising
- Could reach a wider audience

Why you would choose to make your software Open Proprietary:

- Receive an income from selling
- Copyright your code so you can prevent people from modifying and using

Ethical, Legal, Cultural and Environmental Concerns

Issues

- Ethical Issues
- Leal Issues
- Cultural Issues
- Environmental Issues
- Privacy Issues

Specification

Revised

Legal = What is considered right and wrong in the eyes of the law

The Data Protection Act 1998	ⓘ 8 principles relating to the gathering and storage of personal data. ⓘ Data must be kept secure, up to date, cannot sell without permission ⓘ Data subject have the right to see data held about them
Computer Misuse Act 1990	ⓘ To prevent hacking and unauthorised access. ⓘ Covers 3 offences: 1. Gaining unauthorised access 2. Gaining unauthorised access with the intent to commit a rime 3. Gaining unauthorised access and modifying material.
Copyright Design and Patents Act 1988	ⓘ Protects anything that has been created such as images, software, books ⓘ You cannot make copies without permission
Creative Commons Licensing	ⓘ Allows you to legally share media and software with others by giving them permission to use
Freedom of Information Act 2000	ⓘ Allows the public to access information held by organisations

Stakeholder = Individuals or Groups of people who have an interest in an organisation

There will be some overlap between these categories when discussing certain issues. For example Computer Surveillance may be an ethical, cultural and privacy issue.

Ethical

= An **ethical** act is something that is fair and morally right.

Sometimes issues arise that are not covered by any laws.

Examples of ethical issues around computer systems are:

- The Digital Divide = Will everyone be able to afford it?
- Will there be fair charging?
- Are some countries being exploited as a source of cheap labour for call centres and for programming?
- Does the system design promote accessibility for all?
- Is the computer system being misused?
- Cyberbullying and Trolling are problems
- Censorship of the Internet

Cultural

= Issues related to how groups of people with certain beliefs or practices may be affected. Also concerned with changes in society.

- Cyberbullying and Trolling are problems
- Health issues linked to the use of computers. E.g. eye strain, repetitive strain injury, back pain.
- Can improve access to information which can lead to better lifestyles.
- Can lead to stress as we are always 'switched on' such as always checking emails for work
- Some cultures or groups of people may not have access to certain technology.
- Changing the way we work, shop and access information and services.

Environmental

Computer systems can be good for the environment but can also harm it with waste and energy use.

- Computer systems can reduce the number of resources used such as paper
- Creating e-waste. Waste is an issue as computers can contain toxic materials
- Computers use energy. This can be reduced by using modern screens for example.
- Computer systems can lead to more efficient manufacture.
- Computer systems can lead to less fuel being used for transport

Privacy Issues

- Many website required personal information. Is this information private and secure?
- Social media is an issue as lot of personal information and images are published
- Some feel their privacy is being invaded. Images and information being shared online.
- The more data that is store online and on mobile devices increases the risk of it being stolen.

Algorithms
Searching

Standard searching algorithms: Linear and Binary

Specification — Revised

ⓘ **Linear Search** - Items are examined in order from the start of the list until the item is found.

ⓘ **Binary Search** - A type of divide and conquer algorithm where the list is repeatedly split/divided to make searching easier and faster as there are less items to look at

Linear Search	Binary Search
Steps: 1. Look at the first item 2. If this is the item you are looking for then stop as you have found it 3. If not then look at the next item in the list. Repeat steps 2 and 3 until you find your item or get to the end of the list	**Steps:** 1. Sort the list in order 2. Find the middle item in the ordered list 3. If this is the item then stop. 4. If not compare the item to the middle one. If it comes before (is less) than the middle item then get rid of the right side of the list. If it comes after (more than) the middle item get rid of the left side of the list You will have half the list left. Repeat steps 3 and 4 until you get your item
Example: *Given the following list of numbers and you want to find the number 7:* | 2 | 0 | 1 | 7 | 4 | 3 | 5 | 6 | *You will start at 2 and then move to the next items: 0, 1 until you get to 7*	***Example:*** *Given the following list of numbers and you want to find the number 96:* | 6 | 23 | 45 | 55 | 67 | 90 | 92 | 96 | 99 | *You start in the middle with the number 67. The number 96 is greater than 67 so you get rid of the left side of the list and you are left with:* | 67 | 90 | 92 | 96 | 99 | *You go to the middle item again which is 92. (6 is greater than 92 so you get rid of the left side of the list. You then are left with 96 and 99. You can either start with 96 or 99 and you will find your number with one more step.*

Linear vs Binary Search

ⓘ Linear search can be performed when the list is unsorted. With a binary search the list has to be sorted first

ⓘ A binary search is faster with large lists.

Algorithms
Sorting

Standard sorting algorithms: Bubble, Merge and Insertion

Specification — Revised

ⓘ **Bubble Sort** - Compares pairs of items and parses through the list until they are in the correct order

ⓘ **Merge Sort** - Is an example of a divide and conquer algorithm. It splits the list into smaller lists and merges pairs of sub-lists together

ⓘ **Insertion Sort** - Takes each item in turn and places it in the correct place starting with the first item in the list.

Bubble Sort

Steps:

1. Look at the first 2 items in the list
2. If they are in the wrong order swap them
3. Move to the next pair (the 2nd and 3rd items) and repeat step 2
4. Repeat step 3 until you get to the end of the list
   Repeat steps 1-4 until there are no swaps left

ⓘ Not very efficient on large lists
ⓘ For a list of n items you need to do n – 1 passes to sort.

Example:

10	6	5	17	4	9	13	3
6	10	5	17	4	9	13	3
6	5	10	17	4	9	13	3
6	5	10	17	4	9	13	3
6	5	10	4	17	9	13	3
6	5	10	4	9	17	13	3
6	5	10	4	9	13	17	3
6	5	10	4	9	13	3	17

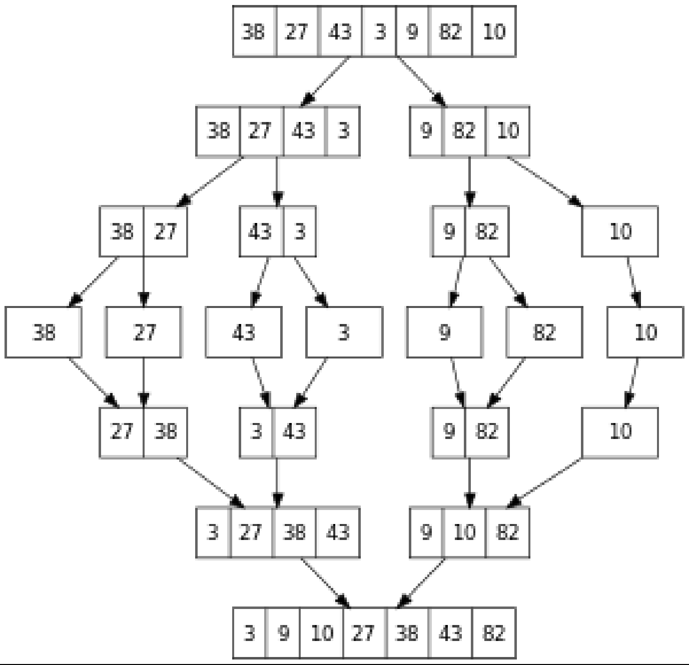
1. *Starting on the left the pairs of items are compared and swapped if needed (in blue).*
2. *We then move to the next pair and repeat until we get to the end of the list.* ***This is first pass****. The largest item will be at the end of the list after the first pass*
3. *Red indicated the item is sorted.*
4. *Steps 1 and 2 are repeated until all items are in order*

Merge Sort

Steps:

1. Split the list in half (the split list are called sub-lists).
2. Keep repeating step 1 on each sub-list until you get lists with 1 item only
3. Merge the pairs of sub-lists together sorting the items in the correct order.
4. Repeat step 3 until you merge all of the sub lists together

Example:



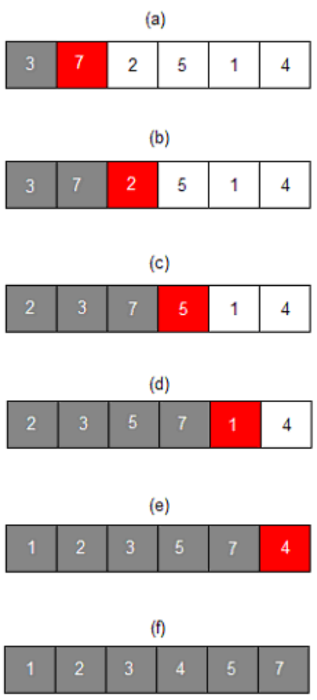
1. The list is split in half and this is repeated until you have sub-lists with only 1 item.
2. The pairs of sub-lists are then merged together in order.
3. This is repeated until they are all merged in order

Insertion Sort

Steps:

1. Look at the second item in the list
2. Compare it to all items before and insert the item in the correct place
   Repeat step 2 until you get the end by moving to the next number and placing it into the correct place.

Example:



1. Each item is taken in turn starting from the left.
2. The red item is taken and inserted into the correct position in the left side of the list.
3. You have 2 lists: The unsorted list to the right and the sorted list to the left of the item.
4. You repeat the steps until it is sorted

Bubble vs Insertion vs Merge Sort

ⓘ Insertion sort uses less memory than a merge sort.
ⓘ Bubble sort is the slowest out of the 3
ⓘ Merge sort is the fastest on large lists out of the 3
ⓘ Bubble sort is a simple algorithm that can be implemented and is efficient to check if a list is already in order.

Algorithms and Programming Techniques

Pseudocode

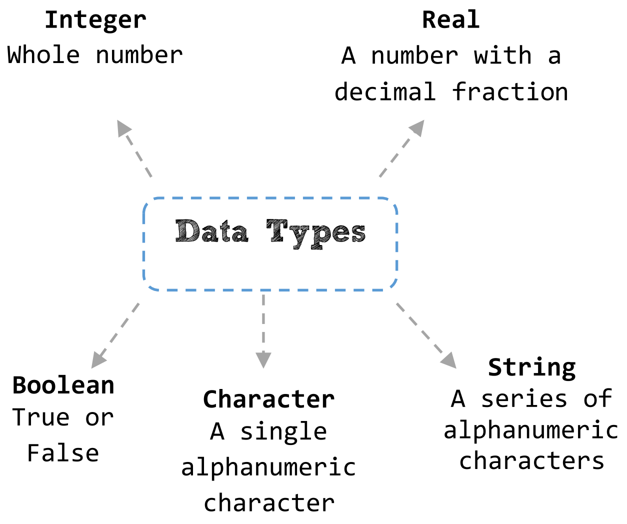
- How to produce Algorithms using Pseudocode and Flowcharts
- Interpret, correct or complete algorithms
- the use of variables, constants, operators, inputs, outputs and assignments
- the use of the three basic programming constructs used to control the flow of a program: sequence selection iteration (count and condition controlled loops)
- the use of data types: integer real o Boolean character and string
- Casting
- The common arithmetic and Boolean operators

Specification — Revised

ⓘ **Variable** = a named memory location which stores a value. The value stored in this space can be used and changed while the program is running. There is usually no value set when declared.

ⓘ **Constant** = a named memory location which stores a value that will not change in the program. Its value is set when declared.

Sequence	Selection	Iteration
Following a set of instructions in order	The path the program takes based on a condition being met. For example, using the IF-ELSE statements.	Repeating a set of instructions. Iteration is another name for a loop



Data Types
Integer
Whole number
Real
A number with a decimal fraction
Boolean
True or False
Character
A single alphanumeric character
String
A series of alphanumeric characters

Mathematical Operators		Comparison Operators	
+	Add	==	Equal to
-	Subtract	>	Greater than
*	Multiple	<	Less than
/	Divide	<> or !=	Is not equal to
MOD	Modulus. Returns the remainder	>=	Greater or equal to
DIV	Returns the integer value from division	<=	Less than or equal to
^	Exponentiation. To the power of		

PROGRAMMING TECHNIQUE	PSEUDOCODE EXAMPLE
INPUT	name = input ("What is your name")
OUTPUT/PRINT	output (name) *or joining a string and variable together* output ("your name is:" + name)

SELECTION
 	
IF..ELSE
	if entry == "a" then
	    print("You selected A")
	elseif entry=="b" then
	    print("You selected B")
	
	else
	    print("Unrecognised selection")
	endif

ITERATION
 	
FOR LOOP
	for i = 1 to 100:
	   output (i)

ITERATION
 	
WHILE LOOP
	count = 0
	while count < 100:
	    output(count)
	    count = count + 1

CASTING (CHANGING ONE DATA TYPE TO ANOTHER)	str(cost) int(score) e.g. str(3) returns "3" int("3") returns 3

Algorithms and Programming Techniques

Pseudocode

- the use of basic string manipulation
- the use of basic file handling operations:
    - open
    - read
    - write
    - close
- the use of records to store data
- the use of SQL to search for data
- the use of arrays (or equivalent) when solving problems, including both one and two dimensional arrays
- how to use sub programs (functions and procedures) to produce structured code

Specification — Revised ☐

STRING HANDLING	
LENGTH OF A STRING	`stringname.length`
GETTING A SUB STRING (PART OF THE STRING)	`stringname.substring(start_position, number_of_characters`
CHANGE STRING TO UPPER CASE	`stringname.upper`
CHANGE STRING TO LOWER CASE	`stringname.lower`

FILE HANDLING	
OPENING A FILE	`myFile = open(“computing.txt”)`

READING FROM A FILE	`myFile = openRead(“computing.txt”)`
	
	`while NOT myFile.endOfFile()`
	`    print(myFile.readLine())`
	`endwhile`
	`myFile.close()`

WRITING TO A FILE	`myFile = `**`openWrite`**`(“sample.txt”)`
	`myFile.`**`writeLine`**`(“Hello World”)`
	`myFile.close()`

ARRAYS/LISTS	
CREATING A LIST	`names = ["Bob", "Alex", "Chris"]`
GETTING AN ITEM FROM A LIST	`names[1]` *this would output Alex*

LOOPING THROUGH A LIST	
	```
	for i in name:
	    output names[i]
	```

LENGTH OF A LIST	`Output (len(names)`

USING A LOOP TO COUNT THE VALUE OF ITEMS IN A LIST	
	```
	scores = [5, 12, 5, 7, 8]
	total = 0
	for i in scores:
	     total = total + i
	```

FUNCTIONS	

FUNCTION
 	
RETURNS A VALUE
	```
	function triple(number)
	     return number*3
	endfunction
	```
	e.g. to call funtion
	y=triple(7)

PROCEDURE
 	
DOES NOT RETURN A VALUE
	```
	procedure greeting(name)
	     print("hello"+name)
	endprocedure
	```
	e.g. to call procedure
	greeting("Hamish")

SQL	
SELECT	
SELECT * FROM Customers	This selects everything (*) from the Customers database.

SELECT *ContactName,Address*
FROM *Customers*
WHERE *ContactName = Mr Smith;*
	This selects the ContactName and Address columns from the Customers table and then specifically looks for a Mr Creosote in the ContactName field.

SELECT *ContactName,Address*
FROM *Customer*
WHERE ContactName LIKE Smi*;
	This selects the ContactName and Address columns from the Customers table and then looks for a something LIKE Smi* in the ContactName field. This is a more open search and will return any value that is like the pattern provided. You can also use these operators:

SELECT * FROM Customers
WHERE Country = 'England'
AND (City = 'Camelot' OR City = 'Palermo');
	You can also use Boolean operators (AND, OR) to refine a search and these can be combined using brackets.

=	Equal
<>	Not equal (!= sometimes)
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
BETWEEN	Between an inclusive range
LIKE	Searcher for a pattern
IN	Specify multiple values

Producing Robust Programs
Defensive Design

- Defensive Design Considerations:
  - Input sanitisation/validation
  - Planning for contingencies
  - Anticipating misuses
  - Authentication

- Producing a Robust program means that it is able to deal with errors and unexpected circumstances so that it can function correctly.
- We can try and achieve this through defensive design.

Input Validation	Input Sanitisation
ⓘ Checking if data is sensible and reasonable (does not check if it is correct) ⓘ Checks it meets given criteria e.g. text must be 8 characters long	ⓘ Cleans the data of any unwanted characters before the data is entered. ⓘ It may remove spaces or symbols from the text before it is input ⓘ Ensures data entered contains only the permitted characters

Types of Validation Checks

Check digit	the last one or two digits in a code are used to check the other digits are correct	Bar code readers in supermarkets use check digits
Format check	checks the data is in the right format	For example a date of birth
Length check	checks the data isn't too short or too long	A password which needs to be 8 letters long
Lookup table	looks up acceptable values in a table	For example gender
Presence check	checks that data has been entered into a field	E.g. you must enter your name when filling out a form
Range check	checks that a value falls within the specified range	E.g. Number of days in month is 1-31

- **Contingency Plan** = We can anticipate how data could be misused or errors that may occur and we can plan for these.
- **Authentication** = check that the user is who they say they are. E.g. authenticate with a user name and password or using a 2 step authentication when a code is generated (using a key device or a message sent) which must be entered to confirm identity. You could also limit the number of login attempts.

Producing Robust Programs

Maintainability and Errors

- Maintainability: Comments and Indentation
- Purpose of Testing
- Types of testing: iterative and final
- How to identify syntax and logic errors
- Selecting and using suitable test data

Specification

Revised

ⓘ Once a program has taken into consideration defensive design it will need to maintain. Easy ways to maintain a program are:

Maintainability = making a program easier to understand and make changes to

Indentation	ⓘ Groups blocks of code together making is easy to read and pick out features ⓘ Blocks of code can be found easily in the future to make changes
Comments	ⓘ Explain the purpose of parts of the code and how they work ⓘ Makes it easier for someone else to understand your code and make changes
Sensible Variable and Sub routine name	ⓘ Makes the program easier to understand when the names used are meaningful and can be followed in the program

Syntax Error	Logic Error
Errors in the rule of the language. Examples: ⓘ Not declaring variables before use ⓘ Not indenting a line of code, missing a colon or bracket. ⓘ Variables not spelt the same as where they are declared	The program may still run but not work as intended. Examples: ⓘ Incorrect output ⓘ Infinite loops ⓘ Incorrect expressions ⓘ Conditions not being met

Producing Robust Programs

Testing

- Purpose of Testing
- Types of testing: iterative and final
- Selecting and using suitable test data

Specification — Revised

- **Testing** = checking that the program works and in the way it was intended to

Iterative Testing	Final Testing
Testing the code as you create it Fixing and making changes as a result Going through this cycle until it meets the requirements	Carried out at the end once it has been written Checks that they requirements have been met Fix any errors

Test Data = Data that is entered when performing the test
Valid/Normal = Test data that represents typical data that should be accepted
Valid Extreme = Test data that is valid but at the end of the range
Invalid = Test data that is outside of the suitable range and should be rejected
Invalid Extreme = Test data that is just outside the valid range
Erroneous: Test data that should be rejected because it is the wrong type

Test Plan = an outline of what is going to be tested. It should cover at least the following:

- Purpose of the test
- Test data
- Expected Result

For example to test the input values in a range where only whole numbers between 1-10 are accepted the following types of test could be carried out.

Test No	What is Being Tested	Test Data	Expected
1	Normal values inside the range are accepted	5 Valid	Value is accepted
2	Values outside the range are rejected	12 Invalid	Value is rejected
3	The last value in the range is accepted	10 Valid Extreme	Value is accepted
4	The values just outside the range are rejected	11 Invalid Extreme	Value is rejected
5	Data of the wrong type is rejected	A Erroneous	Value is rejected

Computational Logic

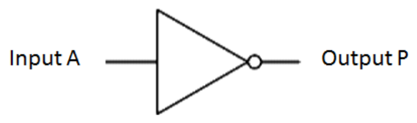
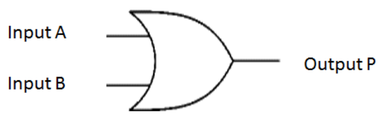
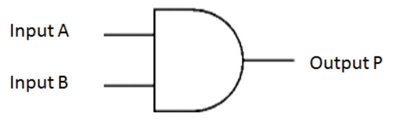
Logic Gates

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

Specification — Revised

- ⓘ A binary number system means that only two digits can be used. These two digits are 0 and 1.
- ⓘ Using only 0 and 1 makes it easier to design the electronic circuits that the computers will use.
- ⓘ Memory and circuits in a computer are made by wiring millions of transistors together. They can make simple logic calculations such as are both inputs a 1. These simple circuits are called logic gates.

There are 3 main types of gates:

NOT	OR	AND
		
ⓘ The NOT gate is a very simple gate – if 0 is input the it outputs 1 and if 1 is input then it outputs 0.	ⓘ The OR gate tells us if one or both of the two inputs are 1 by outputting 1, otherwise output 0	ⓘ The AND gate tells us if both inputs are 1, by outputting 1, otherwise it outputs 0.

Truth Tables

- ⓘ We express this relationship between inputs and output as a truth table.
- ⓘ We use A,B, C...for the inputs and P, Q, R ... as outputs.
- ⓘ Below are the truth tables for the NOT, AND and OR gates

NOT

A	P
0	1
1	0

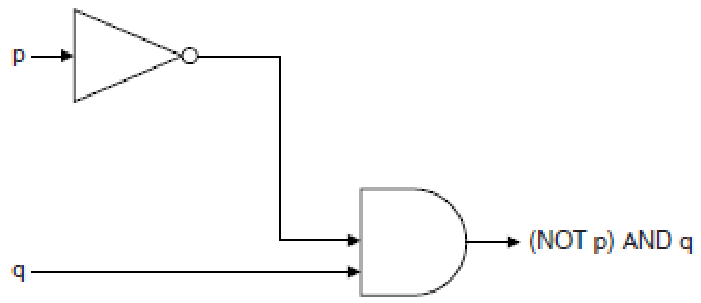
OR

A	B	P
0	0	0
0	1	1
1	0	1
1	1	1

AND

A	B	P
0	0	0
0	1	0
1	0	0
1	1	1

Gates can be combined together to form more complex gates. For example:



Here. We have joined together a NOT gate and an AND gate. The completed truth table would be:

p	q	(NOT p) AND q
0	0	0
1	0	0
0	1	1
1	1	0

You need to first work out the input p which is NOT p. This is in brackets. You do the brackets first

You then combine the result of this with q and show the result of going through the AND gate

Applying Computing Related Mathematics:

Operator	Purpose
+	Addition
-	Subtraction
/	Division
*	Multiplication
Exponentiation(^)	To the power of. E.g. 10^2
MOD	Gives the remainder after division. E.g. 5 MOD 2 = 1
DIV	Gives the integer result of the division. E.g. 5 DIV 2 = 2

Translators and Facilities of Languages

Types of Languages

- Characteristics and purpose of different levels of programming language, including low-level languages

Specification — Revised

ⓘ Machine code is code that is in binary. It is the code the CPU uses when it decodes an instruction and it is specific to a processor.
ⓘ Assembly code uses mnemonics to represent the binary instructions.
ⓘ High level code is understandable to a human being and look more like general English

Machine Code	Assembly Code	High Level Code
ⓘ Low Level Code ⓘ Is what the computer understands (Binary)	ⓘ Low Level ⓘ Uses mnemonics (abbreviations) to represent instructions such as STA for store and LDA to load ⓘ Commonly used for embedded computers	ⓘ Uses commands that are close to structured English ⓘ *Examples include: Python, Java, C++, JavaScript, C#*
ⓘ Processor Specific	ⓘ Processor Specific	ⓘ Portable to different devices
ⓘ Can run straight away and does not need to be translated	ⓘ Needs to be translated to machine code	ⓘ Needs to be translated to machine code

	Low Level	High Level
Advantages	ⓘ Uses less memory ⓘ Can run faster ⓘ Can manipulate hardware components directly	ⓘ Easy to learn and understand ⓘ Easier to find errors/debug ⓘ More portable to different machines
Disadvantages	ⓘ Harder to learn ⓘ Hard to find errors/debug ⓘ Specific to one machine	ⓘ Will run slower ⓘ Takes more memory and slower to run

Translators and Facilities of Languages

Translators

- The purpose of translators
- Characteristics of an assembler, compiler and an Interpreter

Specification — Revised

ⓘ Computers do not understand anything other than machine code. So programs that are written in Assembly or High Level Languages need to be translated into machine code

ⓘ A **translator** is used to translated high level or assembly code into machine code

High level code Machine Code → is translated by a **Compiler** or an **Interpreter**

Assembly Code to Machine Code → is translated by an **Assembler**

Translator	Advantages	Disadvantages
Assembler Convert assembly language into machine code.		
Compiler Converts the ***whole code*** *into* ***object code*** *(an executable version of the program)* before running it. The object code runs independently from the source code and compiler. **Used in distribution.**	ⓘ Customers cannot see the source code when you distribute the program ⓘ Code runs quickly when compiled.	ⓘ Slow to compile ⓘ Can be more difficult to debug as errors are generated at once for all the code
Interpreter Converts the ***code one instruction at a time,*** and executes the instruction before moving to the next. It does not create object code and the source code is run each time. **Mainly used in development.**	ⓘ Easy to debug code ⓘ Code can be developed and tested in stages	ⓘ Interpreter is needed on the target machine ⓘ Slower run time because the program is translated every time it is run.

Object Code = *A translated version of the program (source code) that can be run without the source code*

ⓘ *Web pages are an example of where code is interpreted each time the page loads. The Browser will have an interpreter and run the script that is embedded into the page such as JavaScript.*

Translators and Facilities of Languages

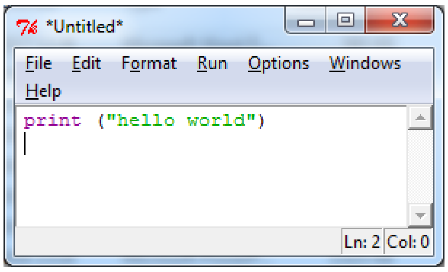
The IDE

- common tools and facilities available in an integrated development environment (IDE):
  - Editors
  - Error diagnostics
  - Run-time environment
  - Translators.

Specification ——— Revised

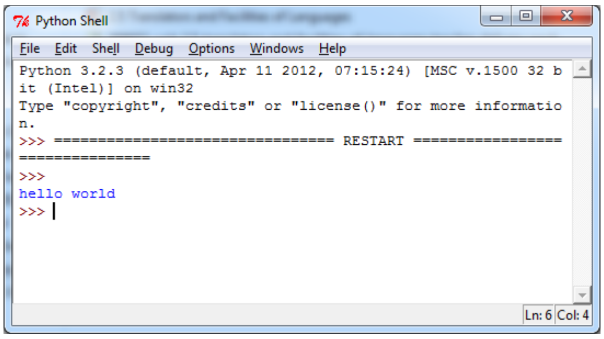
ⓘ The Integrated Development Environment (IDE) has tools and facilities that try to and make it easier to write and debug programs

1. An **editor** which allows you to enter and edit the code. This can have features to make it easier to write your code by using colour coding and automatically indenting your code



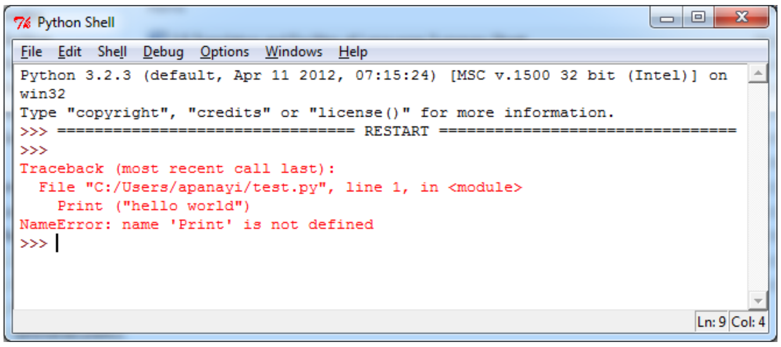
```
print ("hello world")
```

2. **Run time environment**. Allows the code to be run as it is being developed



```
Python 3.2.3 (default, Apr 11 2012, 07:15:24) [MSC v.1500 32 b
it (Intel)] on win32
Type "copyright", "credits" or "license()" for more informatio
n.
>>> ================================ RESTART ================
===============
>>>
hello world
>>> 
```

3. **Translator.** May have a compiler or an interpreter or both to translate the high level code to machine code.
4. **Error Diagnostics**. These identify errors in code and potential problems



```
Python 3.2.3 (default, Apr 11 2012, 07:15:24) [MSC v.1500 32 bit (Intel)] on
win32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
Traceback (most recent call last):
  File "C:/Users/apanayi/test.py", line 1, in <module>
    Print ("hello world")
NameError: name 'Print' is not defined
>>> 
```

Representation of Data

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.

Specification

Revised

Binary is a number system that uses just two symbols, **0** and **1**.

Computers do not understand anything other than machine code. So data and instructions need to be converted and stored in **binary**

- A single zero or one is a **Bit** – **1 or 0**
- 4 zeros and/or ones is a **nibble** – **1010**
- 8 zeros and/or ones is a **Byte** – **10101010**

Common description of bits/bytes	No of bits/bytes
a 1 or a 0 (b)	1 bit
1 nibble	4 bits
1 byte (B)	8 bits
1 Kilobyte (KB)	1024 bytes (1024 x 8 bits)
1 Megabyte (MB)	1024 Kilobytes
1 Gigabyte (GB)	1024 Megabytes
1 Terabyte (TB)	1024 Gigabytes
1 Petabyte	1024 Terabytes

ⓘ *You need to be able to calculate how many bits/bytes in a particular measurement. E.g. How many megabytes in 2 Gigabytes? (2 x 1024 = 2048)*

Representation of Data

Numbers-Binary

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

Specification — Revised

- The **binary** number system uses multiple of two instead of ten (our base 10 number system) for column headings.
- The **denary** number system is our method of counting
- When converting **8 - bit binary** to denary and vice-versa draw a table like the one below with the headings.

128	64	32	16	8	4	2	1

Converting Binary to Denary

We place the binary number in the table and then add up the column values where there is a 1. E.g. convert the number 01010101 to denary:

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

= 64 + 16 + 4 + 1

Answer = 85

Converting Denary to Binary

Start on the left side of the table (largest number) and move along checking if the number is divisible by the column heading. If it is you place a 1 else you place a 0. E.g. convert 56 to binary

- Is the number divisible by 128? The answer is no so we put a 0.
- Is the number divisible by 64? The answer is no so we put a 0.
- Is the number divisible by 32? The answer is yes so we put a 1 and note the remainder which is 24.
- Is 24 (the remainder) divisible by 16? The answer is yes so we put a 1 and note the remainder which is 8.
- Is 8 (the remainder) divisible by 8? The answer is yes so we put a 1 and note the remainder which is 0.
- Is 0 divisible by 4? The answer is no so we put a 0 and note the remainder which is 0.
- We do the same thing for 2 and 1 and place a 0 in the column

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

So therefore 56 in binary is: 00111000

Representation of Data
Numbers-Adding Binary

- How to add two 8 bit binary integers and explain overflow errors which may occur

Specification — Revised

- Adding binary numbers uses the same method as base 10 (denary numbers). We add the values and the value is larger than that column we carry a number to the next column.
- There are **four rules** for adding binary:

1	**0 + 0 = 0** so we write 0
2	**0 + 1 = 1** so we write 1
3	**1 + 1 = 10** so we write 0 and carry 1
4	**1 + 1 + 1 = 11** so we write 1 and carry 1

- Here is an example:

Binary				
	0	1	0	0
	0	1	0	1
+	**1**	**0**	**0**	**1**
Carried values	1			

You can always check your answers by converting them to denary and checking them

- As we are using 8 bits there is a limit on the largest number that can be stored. **Overflow errors** can occur.
- Overflow occurs when we are adding 1 + 1 in the last column. The Carried 1 has nowhere to go and is lost. This causes the wrong answer

Binary								
	1	1	0	0	1	0	1	0
+	1	0	1	1	0	0	1	0
1	**1**	**1**	**1**	**1**	**1**	**1**	**0**	**0**
1						1		

We need a 9th bit otherwise the last bit is lost.

When adding binary numbers do not convert them into denary first, but you can check them at the end

Representation of Data

Numbers-Hexadecimal

- 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

Specification — Revised

- Large binary numbers are difficult for programmers to remember and they want something that is easy to convert from binary.
- Each hex digit represents four binary digits exactly.
- This makes it a quicker way for programmers to write numbers.
- Hexadecimal numbers are based on base 16.
- They have 16 different digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
- We use the letters A to F to represent the decimal numbers 10-15

Hexadecimal	Binary	Decimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
A	1010	10
B	1011	11
C	1100	12
D	1101	13
E	1110	14
F	1111	15

Converting Binary to Hexadecimal

- An 8 bit binary number is split into 2 nibbles.
- Each nibble can store numbers from 0 -15.
  - Example:

10011011

8	**4**	**2**	**1**
1	0	0	1

8	**4**	**2**	**1**
1	0	1	1

Converting Hexadecimal to Denary

- All we do is multiply the numbers by their place values and add them together. For example, take the hexadecimal number 4F.

Place Value	16	1
Hex Digits	3	F
Denary	= 3 x 16	= 15 x 1
	= 48	= 15

- 48 + 15 = 63
- So, 3F is 63 in denary.

Converting Denary to Hexadecimal

1. Covert your denary number into binary in the normal way
2. Split your 8 bit binary number into 2 nibbles
3. Convert each nibble into a hex digit

Or

1. Divide the number by 16 and write down the whole number this is the left most hex digit
2. The remainder is the next HEX digit and this is the digit to the right.

e.g. 172 divided by 16 is 10 remainder 12. 10 is ***A*** *and 12 is* ***C****. So the HEX digits for 172 are* ***AC***

Representation of Data

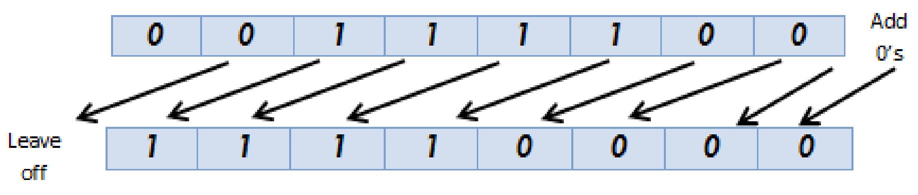
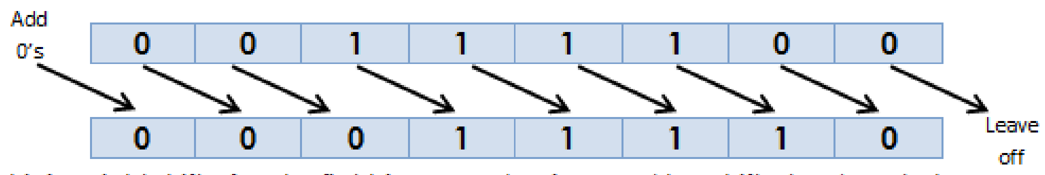
Shifts-Check Digits

- Binary Shifts
- Check Digits

Specification | Revised

A binary Shift moves the Binary number a given number of places to the left or right

ⓘ Moving the binary number to the right will make it smaller (right shift). Moving it to the left will make it larger (left shift)

Example	
Left Shift *Left Shift of 1 is equivalent to multiplying by 2. A shift of 2 is the same as multiplying by 4*	 *This is a left shift of 2. The first binary number is 60 and has shifted 2 places to become 240 (the equivalents of multiplying by4)*
Bits are moved to the left the given number of places. Any gaps are filled with 0's	
Right Shift *Right Shift of 1 is equivalent to dividing by 2. A shift of 2 is the same is dividing by 4*	 *This is a right shift of 1. The first binary number is 60 and has shifted 1 place to become 30 (the equivalents of dividing by 2)*
Bits are moved to the right the given number of places. Any bits that fall of are ignored and any gaps are filled with 0's. There may be a loss of accuracy when dividing as bits may be removed.	

5 014016 150821 >

ⓘ A check digit is an additional digit at the end of a string of other numbers designed to check for mistakes in input or transmission. Calculated using modulo 10.

ⓘ The first 12 digits of the barcode are the unique item number, the 13th is the check digit

ISBN	5	0	1	4	0	1	6	1	5	0	8	2	1
Weight	1	3	1	3	1	3	1	3	1	3	1	3	
Multiplication	5	0	1	12	0	3	6	3	5	0	8	6	
Addition	Add all the numbers												49
Remainder	Find the remainder when divided by 10												9
Subtraction	Subtract the result from 10												1

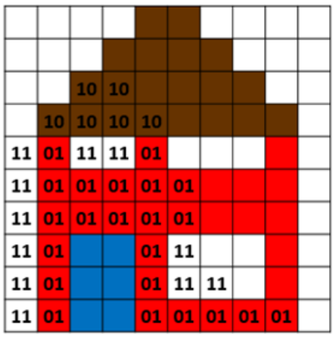
This is an example of how the check digit is calculated using the modulo-10 system. Each digit is multiplied by a weight. All numbers are added together and divided by 10. The remainder is subtracted from 10 to give the check digit

Representation of Data

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.

Specification — Revised



Bitmap Images are represented as a series of Pixels which each pixel stored as a binary code.

- More bits per pixel = more colour combinations
  - 1 bit = 2 Colours
  - **2 bits = 4 Colours**
  - 3 bits = 8 Colours
  - 4 bits = 16 Colours

Pixel	A **pixel (short for picture element) is one specific colour** Each pixel is stored as a binary value
Colour Depth	The number of bits used for each pixel determines how many colours we can use. This is known as the **colour depth.** The more bits per pixel the greater the colour depth and the more bits we need to store the image 4 bits can represent 16 colours, 8 bits can represent 256 colours,
Resolution	The resolution is the concentration of pixels. Usually measure in dots per inch (DPI). The higher the more pixels.
Metadata	Extra information stored about the image. Metadata includes data such as: *[The resolution, Width and height, Colour depth, Exposure, ISO, Aperture, File format]*

The effect of colour depth and resolution on the size of an image file	
A larger **colour depth** and **resolution** = higher quality image + larger file size	A smaller **colour depth** and **resolution** = lower quality image + smaller file size

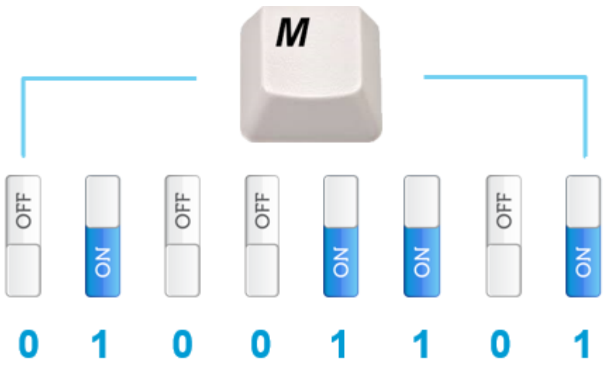
Calculating the size of an image file
Size = resolution x colour depth (in bits) *To get the number of bytes divide the answer by 8*

Representation of Data

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).

Specification — Revised



Every time a character is typed on a keyboard a code number is transmitted to the computer. Characters are stored in binary by assigning a **unique binary code to represent each character**

Character Set	is the group of characters that can be coded
ASCII	Uses 7 bits so can represent 128 (2^7) characters
Extended ASCII	Uses 8 Bits so can represent 256 (2^8) characters
Unicode	Uses 16 bits so can represent 65,536 (2^{16}) characters

Problem with ASCII and Use of Unicode

With all the languages and characters/symbols in the world ASCII is not sufficient enough to represent all of these (only 128 or 256 characters). Unicode uses 16 bits and allow a much larger character set (65,536 or 2^{16}).

ASCII and Pure Binary

- Numbers as Characters have a different ASCII code compared to their actual (pure) binary representation. For example the character **1** has the ASCII code 0110001 which is 48 in Denary. The **pure binary** for 1 is 000001.

- You cannot do arithmetic on ASCII characters representing numbers – they must first be converted to pure binary numbers.

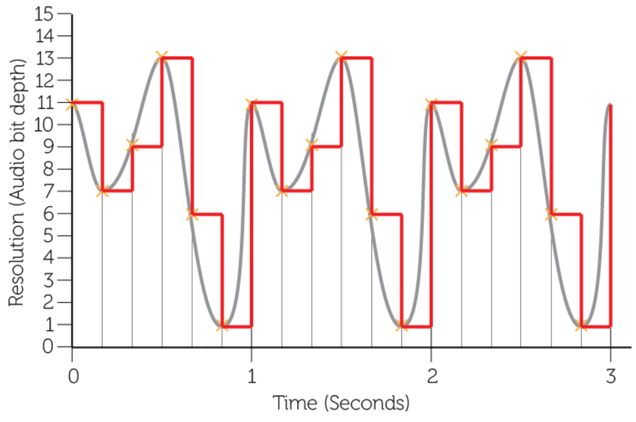
Representation of Data

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.

Specification

Revised



Sampling – Measuring the amplitude (height of the wave) at regular intervals. Store each measurement in binary. The more bits used for the bit rate the more accurate the measurement

Bit Rate	The number of bits used to store each sample. – The more bits used the better the accuracy of the sound file.
Sampling Frequency	The time period between taking samples/measurements. – The more frequent the sound is sampled, the better the quality of playback
Sample Size	The length of the recording sampled

Size and Quality of the Sound File
ⓘ *Recording quality improves:*   – *the more frequently we sample the sound*   – *the more accurately we record the wave height (larger bit rate)*   ⓘ *Increasing sampling rate and resolution means recording more data points which involved more data so the file size is therefore larger*

Size of a Sound File
Size = Bit Rate (bits) x Sample Frequency x sample size (duration in secs)

Representation of Data

Compression

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

Specification ——— Revised

Bitmap Images are represented as a series of **Pixels** which each pixel stored as a binary code.

- More bits per pixel = more colour combinations
  - 1 bit = 2 Colours
  - 2 bits = 4 Colours
  - 3 bits = 8 Colours
  - 4 bits = 16 Colours

Pixel	A **pixel (short for picture element) is one specific colour** Each pixel is stored as a binary value
Colour Depth	The number of bits used for each pixel determines how many colours we can use. This is known as the **colour depth.** The more bits per pixel the greater the colour depth and the more bits we need to store the image 4 bits can represent 16 colours, 8 bits can represent 256 colours,
Resolution	The resolution is the concentration of pixels. Usually measure in dots per inch (DPI). The higher the more pixels.
Metadata	Extra information stored about the image. Metadata includes data such as: *[The resolution, Width and height, Colour depth, Exposure, ISO, Aperture, File format]*

The effect of colour depth and resolution on the size of an image file	
A larger **colour depth** and **resolution** = higher quality image + larger file size	A smaller **colour depth** and **resolution** = lower quality image + smaller file size

Calculating the size of an image file
Size = resolution x colour depth (in bits) *To get the number of bytes divide the answer by 8*