Fundamentals of data representation

Number systems

- Natural numbers - $\mathbb{N}$ = {0, 1, 2, 3, ... }
- Integer numbers - $\mathbb{Z}$ = { ..., -3, -2, -1, 0, 1, 2, 3, ... }
- Rational numbers - $\mathbb{Q}$ Any number that can be represented as a fraction
- Irrational numbers - Any number that **cannot** be represented as a fraction
- Real numbers - $\mathbb{R}$ is the set of all 'possible real world quantities'. This includes natural, rational and irrational numbers
- Ordinal numbers - Describes the numerical position of objects

Number Bases

- Binary (Base-2)

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

- The unsigned binary number represented here is 262
- For all binary numbers the minimum value that can be represented is 0 and the maximum is $2^n - 1$
- The -1 is there since 0 is included in the counting
- Hexadecimal (Base-16)

Decimal	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
Hexadecimal	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

- To convert from binary to hexadecimal, split the number into group of four, then convert them to hexadecimal
- Binary to hex : 0011101011111001 → 0011 1010 1111 1001 → 3AF9
- Hexadecimal is often used in favour of binary since it is easier for a human to read and remember. One byte can be represented in two digits in hexadecimal instead of eight in binary.
- Hexadecimal is commonly used for memory addresses and RGB colour codes

Units of information

- Bit - 1 or 0
- Byte - 8 bits
- Nibble - 4 bits

- Number of values that can be represented in n bits is 2^n where is largest value is $2^n - 1$
- ∴ 8 bits can represent 2^8 (256) values in the range of 0 ... $2^8 - 1$ (0 - 255)

The following are the units to count bytes in base-10

- Kilobyte - KB - 1000 B
- Megabyte - MB - 1,000,000 B
- Gigabyte - GB - 1,000,000,000 B
- Terabyte - TB - 10^{12} B

The following are the units to count bytes in base-2

- Kibibyte - KiB - 2^{10} B
- Mebibyte - MiB - 2^{20} B
- Gibibyte - GiB - 2^{30} B
- Tebibyte - TiB - 2^{40} B

Binary number system

Addition

- 0 + 0 = 0
- 0 + 1 = 1
- 1 + 0 = 1
- 1 + 1 = 0 (carry 1)
- 1 + 1 + 1 = 1 (carry 1)

- The result of an addition may overflow. This could cause a negative result

Multiplication

- Shift one place to the left for every digit
- If multiplying by 1, copy the top number
- Ignore and shift if multiplying by 0

Two's complement

- Common method to represent a negative number
- The most significant bit is negative

- To convert: start from the **right**, leave all digital up to and including the first 1, then flip all digits

	-128	**64**	**32**	**16**	**8**	**4**	**2**	**1**	
Unsigned	0	1	0	0	0	0	0	1	= +65
Signed	1	0	1	1	1	1	1	1	= -65

- The range of a two's complement number is - (2^{n-1}) ... 2^{n-1} - 1
- It is n-1 since the MSB is used as the sign bit

- Two's complement can be used for subtraction by converting the number you want to subtract to a negative then add the two numbers

Fixed point binary

- This is used to represent numbers with a fractional part

8	4	2	1	.	1/2	1/4	1/8	1/16
0	1	0	1	.	1	1	0	0

- 0101 1100 represents 5.75 in fixed point binary

Information coding systems

- A character code is a unique number that represents a character

ASCII

- A method to represent characters
- Uses 7 bits for 128 combination
- This is enough for all english characters and other common printed and non-printed characters

- The character 'A' is represented by the character code 65 in decimal and 1000001 in binary
- This character code cannot be used for arithmetic since the numerical value is not the not the same as the character value

Unicode

- There were many incompatible coding systems
- Unicode-16 allows 16 bits per character
- UTF-32 has over 1 million combinations

- Unicode requires more bits than ASCII ∴ increases file size

Parity bits

- These are addition bits to check if bits were sent correctly
- Odd or even parity can be used. This represents an odd or even number of 1's in the bit pattern
- With ASCII characters, the 8th bit is the parity bit

- eg. The letter 'R' in ASCII is 1010010. With even parity it would be 11010010. There are four 1's

Majority voting

- Each bit is sent multiple times
- You accept the most common
- If you receive: 001 110 010
- You would accept 0 1 0
- A downside is that more data is sent. In this case 3x the data is sent.

Check digit

- Additional digit at the end of a string of numbers

- If once number changes the check digit will change
- Commonly used on barcodes and ISBN numbers

Bitmap graphics

- A pixel is a picture element
- Each pixel has a binary which represents a single colour
- Bitmap is made of picture elements

- WxH in pixels is the resolution
- This is not linked to the size of the image since the pixels can be any size
- Increasing the number of pixels increases the sharpness of the image

- Pixels per inch is the pixel density

- Colour depth is the number of bits per pixel
- This determines the number of colour combinations.

- Image size = number of pixels x colour depth

- Metadata - data held within the image to allow it to be interpreted corrected. Eg. date last modified, file size, file format, colour depth

Vector graphics

- Not stored as pixels
- Stored as a formula for shapes and objects
- Stores details with which software can draw the image

- Details are held in a drawing list:
    - Shape = circle
    - Centre = x,y
    - fill = blue
    - border = black

- The image is redrawn when resized ∴ no quality loss
- Smaller file size for geometric shapes since less data is stores
- This means they are often used as logos since they can be infinitely scaled and the smaller file size allows for faster transmission
- Vectors are bad for photographs

Digital sound

- Sample rate - samples taken per second
- Bit depth - bits per sample

- Sound is an analogue continuous wave, that muse be converted to digital and discrete data

- Amplitude is measured and recorded at given intervals. More samples = more accurate sound
- Increasing the bit depth increases the levels of amplitude that can be measured which increases the sound quality

- Increasing the sample rate leads to smooth audio but a larger file size

- Nyquist theorem - for an accurate recording, the sample rate must be double the original frequency
- The average human hearing has a maximum frequency of 22,000 ∴ 44,000 is the standard sample rate for a high definition recording

- To calculate the size of a sound file:
    - The number of samples per second x the number of bits per sample x the length of the sample in seconds

- The following steps are taken in an analogue to digital converter (ADC)
    1. Regular samples are taken of the analogue signal
    2. Amplitude of each sample is approximated to a digital value
    3. The value is then encoded as a binary value in a fixed number of bits
    4. Binary is then output as a digital signal

- The following steps are taken in a digital to analogue converter (DAC)

    1. Binary data is converted to digital in volts
    2. The digital signal is then smoothed to produce an analogue signal

- MIDI - Musical Instrument Digital Interface

    - A MIDI controller carries event messages that specify the note, pitch, instrument etc.
    - It is a list of instructions to synthesise sound
    - Can use up to 1000x less disk space
    - Easily edited

Data compression

- Compression is used to use less storage space and increase transmission speeds

- Lossy compression removed non essential information
    - This reduces the files size but leads to a loss in quality
    - MP3 is a common lossy form of compression used for audio files. It removes sound that is out of the hearing range and that may be drowned out by other sounds
- Lossless compression records patterns in the data are compresses in a way that can be reversed
    - No data is lost ∴ useful for program files
    - File sizes may be larger than lossy compression

- Run length encoding is a form of lossless compression where sequences in which the same data value occurs consecutively the data elements are stored as a single item with a value and count

- Dictionary based compression
    - A dictionary is stored along with the text. An algorithm searches through text to the find suitable entries for the dictionary and translates
    - With large amounts of text, the size of the dictionary becomes insignificant
    - Completely lossless

Data encryption

- Encryption is the transformation of data from one form to another to prevent third parties from being able to understand it
- The original data is known as plaintext
- The encrypted data is known as ciphertext

The Caesar cipher

- This is a type of substitution cipher
- The letters of the alphabet are shifted along a number which is the key.
- eg. with a key of 5, the letter A becomes F, B becomes G etc.
- This is a very weak cipher. It can be easily brute forced since there are only 25 possible keys
- For longer messages cryptanalysis could be used by finding the most common letter

Cryptanalysis and perfect security

- Ciphers that use non random keys are open to cryptanalytic attacks and can also be brute forced given enough time and resources
- Random numbers that are generated by an algorithm can be broken since they are not actually random
- To get truly random numbers, the sequence must be collected from a physical and unpredictable phenomenon such as radioactive decay

The Vernam cipher

- The Vernam cipher is an implementation of a class of ciphers known as one-time pad ciphers
- The one-time pad must be equal to ore longer than the plaintext, be truly random and be used only once
- The key must be shared in person and destroyed after use
- If the key is truly random no amount of cryptanalysis will produce meaningful results since the distribution of characters will be random
- The binary representation of each character and the corresponding character on the one-time pad are XOR'd

Plaintext : M	Key: +	XOR : f
1	0	1
0	1	1
0	0	0
1	1	0
1	0	1
0	1	1
1	1	0

Fundamentals of computer systems

Hardware and software

- Hardware is the term for the physical parts of a computer used for input, output and storage
- Software is the term for the programs that run on the hardware

System software

Operating system

- The operating system provides a user interface between user and the hardware
- The operating system has the following functions:
    - Memory management
        - Allows you to do several tasks at once. Each process is allocated an area in memory which is controlled by the OS. The hard disk may be used as virtual memory when the RAM is not enough
    - Processor scheduling
        - This is how the OS allocates processor time for all applications running. A single core processor can process small parts of many tasks in turn to give the appearance of **multi-tasking**. This is controlled by the scheduler.
    - Backing storage management
        - The OS keeps a directory of where files are stored so that they can be accessed quickly and the areas that are free for new files are found quickly
    - I/O control
        - This ensures that peripherals are allocated to processes without conflicts. Different applications require different input and output devices throughout their operation.
    - Interrupt handling
        - This is how the OS handles a signal from an application or peripheral that causes it to stop processing its current list of instructions. The OS detects it and sends and calls the relevant handler

Utility programs

- These are programs that are used to configure, optimise and maintain the computer
- For example:
    - Virus scanner
    - Disk defragmenter
    - System monitor
    - File manager

Translators

- All translators convert source code to machine code

- Assembler

    - assembly is a low level language which means it is similar to the processor's instruction set
    - Each processor has a different instruction set
    - The assembly turns the assembly into code that is executable on the processor

- Compiler
    - Converts the high level source code into executable code
    - It first scans the code to detect errors
    - The object code that is produced can then be saved
- Interpreter
    - Interpreters go through source code a line at a time, translate then execute
    - Also scans for errors

Programming language classification

- Opcode - This is the instruction that the processor will execute. This may be an operation such as LOAD or ADD
- Operand - This is the data to be operated on or the address for the data. This can take two forms
    - Immediate addressing - Where the operand is the actual value to be operated on
    - Direct addressing - Where the operand is the memory address of the value to be operated on

Machine Code

- Machine code - This is the binary that the computer can understand
- It is a low level language since the code reflects how the computer carries out the instruction
- A typical instruction consists of an opcode and operand
- The length is often 32 bits
- eg. 0000 may represent a LOAD instruction and 1000 may represent HALT

Assembly language

- A one to one mapping of machine code
- Opcodes are replaced with mnemonics which indicate what the opcode represents
- The operand is replaced by a decimal or hexadecimal number

High-level languages

- High-level languages allow programmer to think in terms of algorithms instead of small steps and details such as where a variable will be stored in memory
- Imperative high-level languages are where instructions are executed in a programmer defined order which describes how to solve a problem

Program translators

- Different translators are used depending on the source code
- Assembler - Use to translate assembly to machine code. It is a 1 to 1 translation
- Compiler
    - Converts high-level source code to executable code
    - It scans the code for errors

    - Different platforms require different compilers since the machine code produced is hardware specific

- Interpreter
    - Looks at the source code a line at a time then translates and executes
    - If there was on error the program would run up until the error is met
    - It can also scan for syntax errors

- Advantages of a compiler
    - The executable code that is produced can be saved an run without compiling
    - The executable can then be distributed and executed without a compiler
    - Source code cannot be read after it has been complied ∴ it is more secure

- Advantages of an interpreter
    - Useful for development. it does not have to be recompiled for each change
    - Easier to test and debug small parts of programs

- Disadvantages of an interpreter
    - Runs slower than a compiled program since it has to translate and run at the same time
    - Each statement has to be translated each time
    - A loop of 10 statements performed 20 times would have all statements interpreted 20 times

- Bytecode
    - It is an instruction set executed using a virtual machine
    - The virtual machine emulates the architecture of a computer ∴ faster to execute than an interpreted language however it is still slower than a natively compiled program
    - Guards form malicious programs since it is not run directly on the hardware

Fundamentals of computer organisation and architecture

- A computer system is made up of both internal and external components
- Internal components include
    - processor
    - main memory
    - address, control and data bus
    - I/O controller

The processor

- The processor is made up of the following components

Arithmetic Logic Unit (ALU)

- Performs arithmetic and logical operations on the data. Such as ADD, SUBTRACT, MULTIPLY and DIVIDE.

- Can also shift bits to the left or right within a register
- Can carry out boolean logic and compare values using AND, OR, NOT, XOR

Control unit

- Controls and coordinates the actions of the CPU
- Directs the flow of data between the CPU and other components
- It accepts instructions, breaks down the steps required to process the instruction, manages the execution and store the resulting data back in memory and registers

The system clock

- The clock synchronises the CPU's operations by switching between 0 and 1 billions of times per second
- The clock speed is measured in Hz
- Some CPU operations take multiple clock cycles

General-purpose registers

- These are small and very past areas of memory inside the processor that is used to hold data that is being operated on
- The accumulator is where the result of a calculation or logical expression is held

Dedicated registers

Program counter

- Holds the address of the next instruction to be executed
- This may be the next area in memory or the target of a branch operation

Current instruction register

- Holds the instruction currently being executed

Memory buffer register

- Use to temporarily store data read or written to memory
- Also called the memory data register

Status register

- Contains bits that are set or cleared based on the result of an instruction
- eg. A bit may be set if an overflow occurred
- A bit may be set to show if the result of the last instruction was negative, zero or caused a carry

The fetch-execute cycle

Fetch

- Contents of the PC to MAR
- Transferred to main memory via the address bus

- Contents of the addressed location to MBR
- Transferred via data bus
- PC incremented
- MBR to CIR

Decode

- The instruction hed in the CIR is decoded
- The instruction is split into an opcode and operand
- The opcode determines the type of instruction
- Additional data is fetch if necessary

Execute

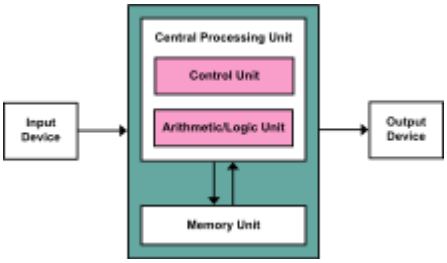
- The instruction is executes
- The result is stored in the accumulator, a general purpose register or main memory

Factors affecting processor performance

- Clock speed - Increasing the clock speed means more instructions executed per second
- Bus width - More data is transferred per clock cycle
- Word length - Related to the bus width. Higher bus width can process more data at once
- Multiple cores - Many instructions can be fetched and executed in parallel
- Cache - Data that may be needed for the next instruction in held in the cache which can be accessed faster than main memory Level 1 cache is fast but small Level 2 is fairly fast and medium sized Some CPUs have Level 3 cache

Memory and the stored program concept

- The stored program concept : machine code instructions are fetch and executed serially by a processor that perform arithmetic and logical operations
    - Instructions are stored in main memory which are then fetch and executed by the processor. Programs can be moved in and out of memory
- Most computers run on the Von Neumann architecture which is where data and instructions are held in the same memory



- Some computers run on the Harvard architecture
    - Different buses for data and instructions. Both stored in different memory
    - Data and instructions can be fetch in parallel ∴ instructions handled more quickly
    - Used in embedded systems and digital signal processing

Address, control and data bus

- A bus is a set of parallel wires connecting two or more components of a computer
- The address, control and data buses connect the processor to main memory

Control bus

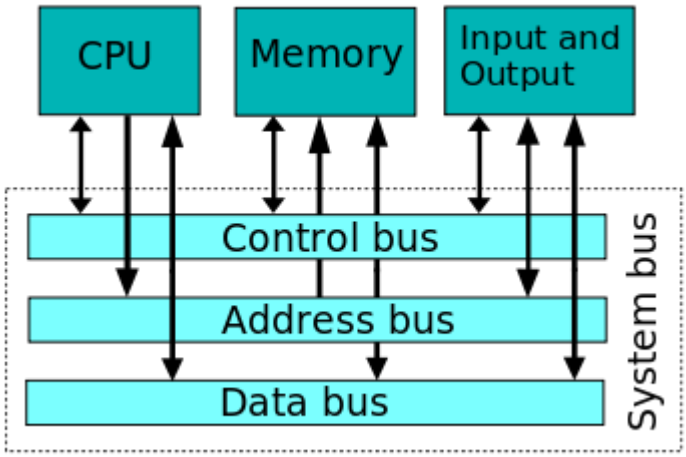
- A bi-direction bus that transmits command, timing and specific status information between components
- Controls the flow of data between the processors and other components
- Achieved by syncing signals and having control of access to data

Data bus

- A bi-directional bus to transfer data between the processor and main memory
- The width of the bus is key to overall performance of the CPU

Address bus

- A unidirectional bus from processor to main memory
- It carries the address of the next instruction or data item
- An 8 bit address bus would be able to address 2^8 memory locations



I/O controller

- Interfaces between and I/O devices and the processor
- Each device has a separate controller which connects to the control bus
- They receive I/O requests from the processor then send device-specific control signals

- They also manage the flow of data to and from the devices
- The controller consists of three parts:
    - An interface for connection of the controller to the system or I/O bus
    - Data, command and status registers
    - An interface to connect the controller to the device's cable

I/O devices

Barcodes

- There are two types of barcode
    - Linear(1D) and 2D(QR)
- There are four types of readers : pen type, laser scanner, CCD and camera based

Pen type

- The pen is dragged across the bars at an even speed
- The photo diode measure the intensity of light reflected back from the light source
- This generates a waveform to measure widths of bars and spaces
- Once converted to digital, the waveform is identical to the bars
- The bats are then encoded using binary

- This is the most durable type of barcode scanner
- Must be in direct contact with the barcode
- Portable

Laser scanner

- Works the same as a pen scanner
- The laser reflects off a mirror allowing it to be read in many positions

CCD

- Uses hundreds of light sensors which measure the light intensity directly in front of them
- The voltage pattern produced is identical to the barcode generated by reading voltages in a row

Camera based

- Uses camera and image processing
- Can be read off any surface and even if the barcode is in a bad condition

Digital cameras

- Uses CCD sensor
- When the shutter opens it projects an image onto the sensors
- The sensors measure the brightness of each pixel
- This is converted to electricity and stores the charge as binary data

- CCD is higher quality and more reliable

- However uses 100x the power of CMOS

RFID

- Reader sends a wave to the tag
- The tag is then energised by the waves
- The transponder then transmits data to the reader

- Works without a line of sight up to 300m
- Can pass stored data from the tag to the receiver and vice versa

Laser printer

- The drum is coated in a positive charge
- Printer generated a bitmap from the data
- The laser is shone at the drum via a rotating mirror
- The laser is modulated to remove charge where there should be toner
- The toner is given a positive charge
- The toner is attracted to the drum
- Toner is transferred from the drum by rolling over paper
- The toner is fused to the paper by heated rollers
- This is then repeated 4 times for colour (CMYK)

Storage devices

- Secondary storage is not directly accessible by the processor ∴ slower access times than RAM
- Secondary storage retains contents without power (non-volatile) unlike RAM which is volatile

Hard disk

- Iron particles are polarised to 0 or 1
- A R/W head moves across the spinning disk to access different tracks and sectors
- Data is read or written to the disk when the disk passes under the R/W head

Solid state drive

- Uses NAND flash memory
- Data is stored in floating gate transistors
- Floating gate transistors do not lose state when no power is applied
- Cannot read/write individual bits

SSD	**Hard disk**
Lower power consumption	Lower cost per unit storage
Faster access times	Higher capacity drives available
Less vulnerable to physical damage	Less concern about maximum number of write cycles
Noiseless operation	
Less heat generated	

Optical disk

- Three types - ROM, R, RW

- Data bits are recorded by burning a pit, making that area less reflective
- Change from a pit to land indicates a 1 with everything else being a 0
- High powered laser burns pits
- Lower powered laser used to read

- CD-ROM holds 650MB
- Blu-Ray can hold up to 50GB

- Disks may not be readable in the future for the following reasons
    - No hardware available to read
    - Scratched
    - No software available

Fundamentals of communication and networking

Serial and parallel

- Serial
    - bits sent one bit at one time over a single wire
    - High transfer rates can be achieved
- Parallel
    - Several bits sent simultaneously over parallel wires
    - Used in integrated circuits
    - Wires have different properties, bits travel at different speeds, skew can develop
    - Only suitable over short distances

Advantages of serial

- Serial is reliable over much longer distances than parallel
- Interference between parallel lines leads to corrupted words which means data will have to be sent again
- Simpler and smaller connectors which means a lower cost
- Because of the lack of interference at high frequencies, signal frequency can be higher meaning there is a higher net transfer rate even though less data is transmitted per cycle

Bit and baud rate

- Bit rate
    - Speed at which data is serially transmitted
- Baud rate
    - Rate at which signal changes

- When there are only 2 voltage levels used, the bit rate and baud rate are the same
- Bit rate = baud rate x bits per signal
- bit rate and baud rate can be different if a signal contains one or more bits

Bandwidth

- Range of frequencies that a transmission median can carry
- Increasing bandwidth increases the amount of data transferred per unit time
- There is a direct relationship between bandwidth and bit rate

Latency

- Time delay between the moment of transmission of the first packet and when it is received
- High latency will lead to a large delay

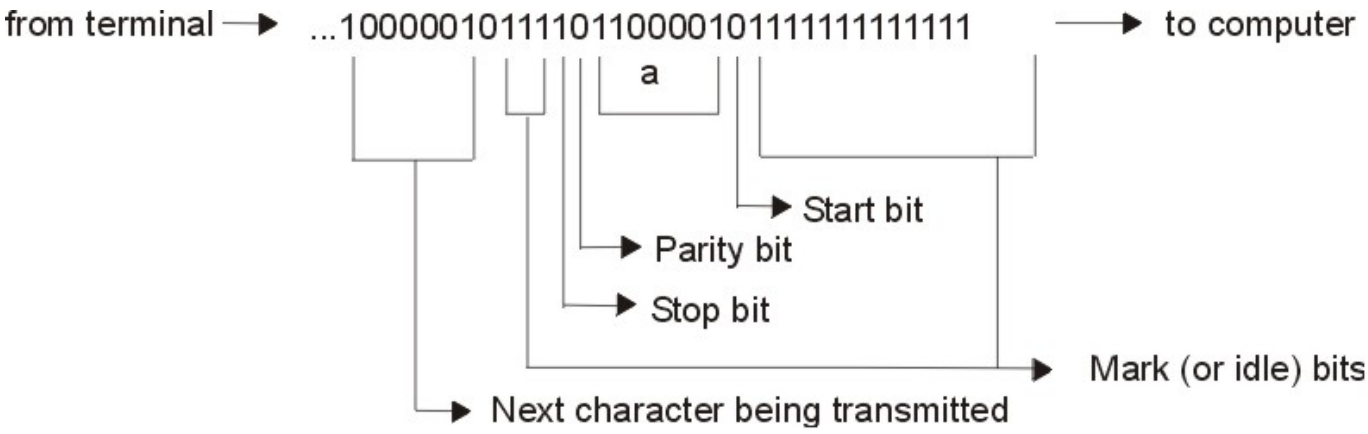
Synchronous transmission

- Data is transferred at regular intervals controlled by a clock signal
- Used in parallel communication since the clock allows for constant, reliable transmission of time sensitive data

Asynchronous transmission

- Data is sent intermittently rather than as a continuous stream
- Start bit synchronises the clock in the receiver
- Receiver's baud rate must be the same

- Parity bit is used for error checking
- Start and stop bit must be different



Protocol

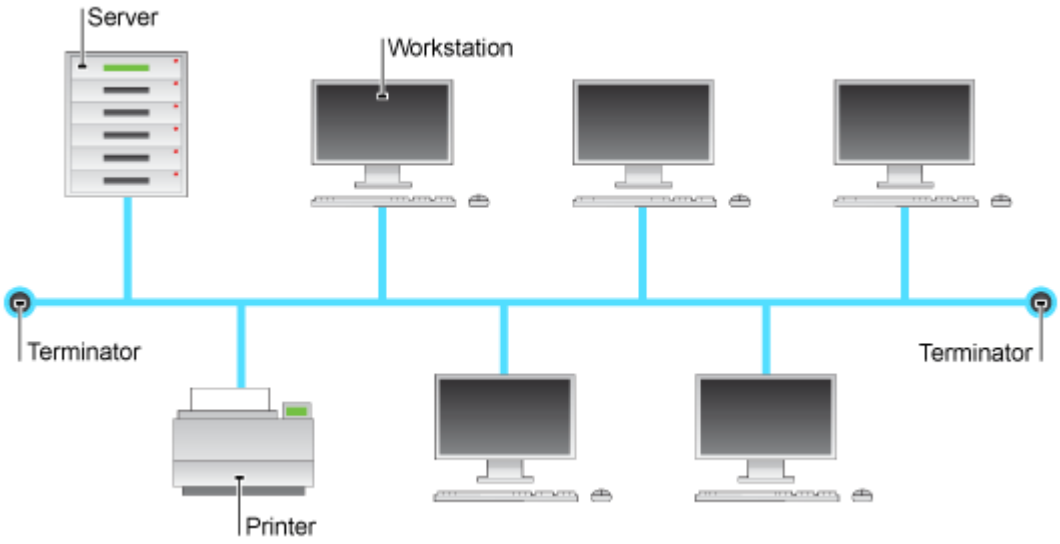
- A protocol is the rules for communication between devices
- Allows devices to be networked
- Protocols may include
    - Types of physical connections
    - Mode of transmission
    - Speed
    - Data format
    - Error detection and correction

Network topology

- Local areas network
  - LAN. Number of computing devices on a single site connected by cables
  - Users can communicate, share data and hardware

Bus topology

- Computer connected to a single cable
- Ends of cable connected to a terminator



Advantages of bus topology

- Inexpensive
  - Less cable needed than in a star topology
  - No additional hardware required

Disadvantages of bus topology

- If the main cable fails, network data can no longer be transmitted
- Performance degrades with heavy traffic
- Low security
  - All computers see all network transmissions

Physical star topology

- A central node (switch/server) acts as a router

Advantages of a star topology

- Faults isolated
  - Cable failing only affects one station
- Consistent performance under heavy load
- No issues with data collisions
- More secure

    - Messages sent directly to central node
- Easy to add new stations

Disadvantages of a star topology

- More cable required
    - More costly
- Network transmission is not possible if the central node fails

Operation of a star network

- A central node such as a switch records the MAC address of each device to identify where to send data

MAC address

- Every network interface card (NIC) has a unique MAC address.
    - A MAC address is 48 bits in hexadecimal

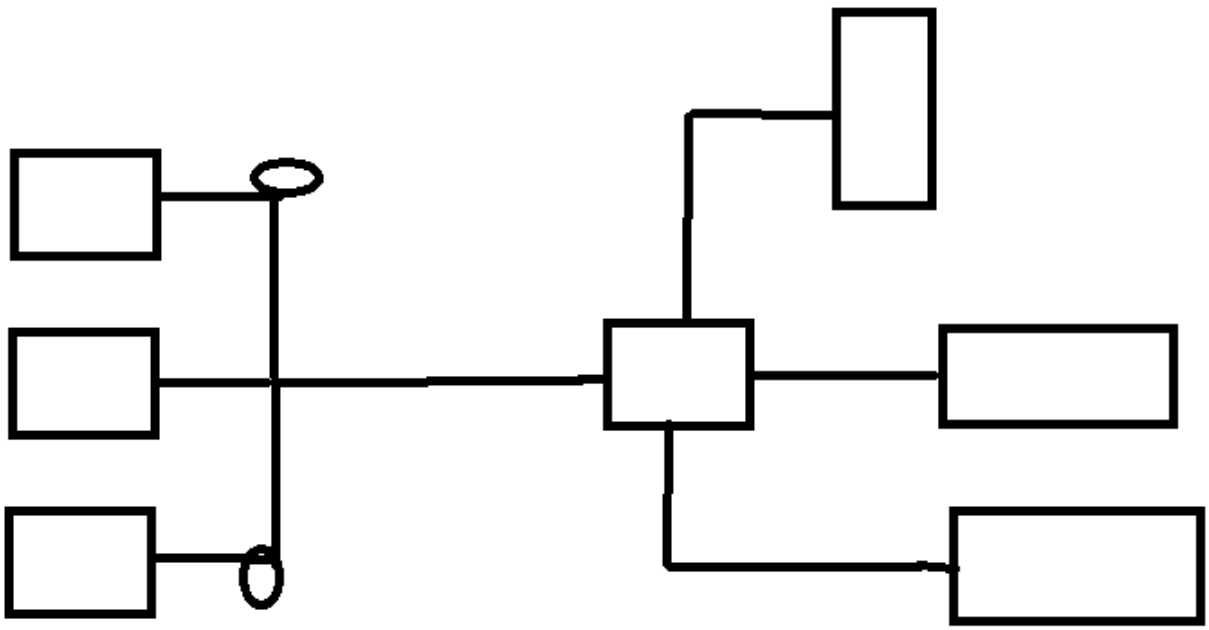
Physical vs logical topology

Physical

- The physical topology is the actual layout and design of the network ie. the shape of the wiring

Logical

- The shape of the path that the data travels in
- How components communicate across the physical topology



- On the left, computers are connected in a physical bus connection
- Ont he right, computers are connected in a physical star network
- However, logically, the right hand side can be arranged as a bus network

Operation of a logical bus network

- Data cannot be transmitted in both directions
- Every station receives all network traffic
- Traffic from each station has equal transmission priority

Client-server networking

- Clients connect to a central computer called a servers
- There are different servers to handle different content
    - File server holds and manages client's data
    - Mail server manages email system
    - Web server manages requests to access the web
- A client makes a request to the server which then processes it

Advantages of client-server

- Better security
    - All files stored in a central location and the access rights are managed centrally
- Backups are done centrally
- Data and hardware can be shared

Disadvantages of client-server

- Expensive to install and manage
- Staff needed to maintain and run the server and network

Peer to peer network

- Individual computers connected to each other

Advantages of a peer to peer network

- Easy and inexpensive setup
- Resources can be shared
- Easy to maintain
- Peer to peer is often used for online piracy

Peer-to-peer

Client/Server

Wi-Fi

- Local areas wireless technology that enables you to connect a device to the internet via a wireless access point (WAP)

Components required for wireless networking

- Wireless network adapter
- A computer and interface controller is called a station
- All stations share a single frequency
    - Each station is always tuned to the frequency to receive transmission
- To connect to the internet, the WAP usually connects to a router

Security of a wireless network

WPA

- Wi-Fi Protected Access and WPA2 are security protocols and security certification programs
- WPA2 is build into NICs
    - This provides strong encryption
    - A new 128 bit key is generated for each packet

SSID

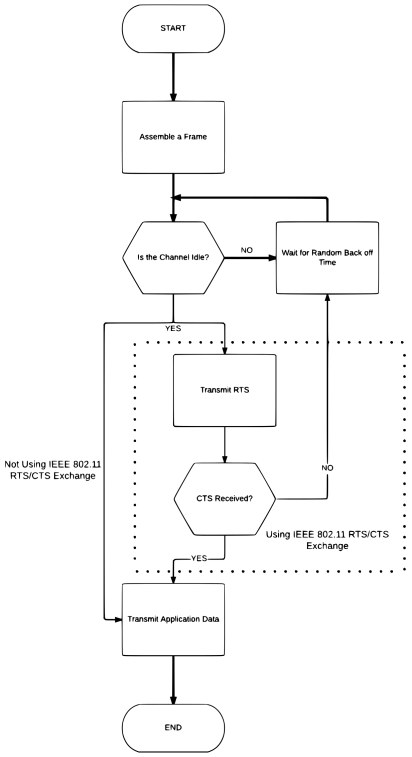
- Each wireless network has a Service Set ID (SSID)
    - This is the human-readable name for the network
    - It helps identify the network
    - Broadcasting the SSID can be disabled
        - This hides the network for people looking to connect to a local network
        - Which can be seen as increasing the security of the network

Whitelists

- A MAC address whitelist controls who is allowed on a network
- If a MAC address is not on the whitelist, the device will not be able to connect

CSMA/CA

- Carrier Sense Multiple Access / Collision Avoidance



CSMA/CA with RTS/CTS (Shown in dotted box)

- If no other note is transmitting, the stations send a Request to Send (RTS) signal
- WAP sends a Clear to Send (CTS) if and when the channel is idle

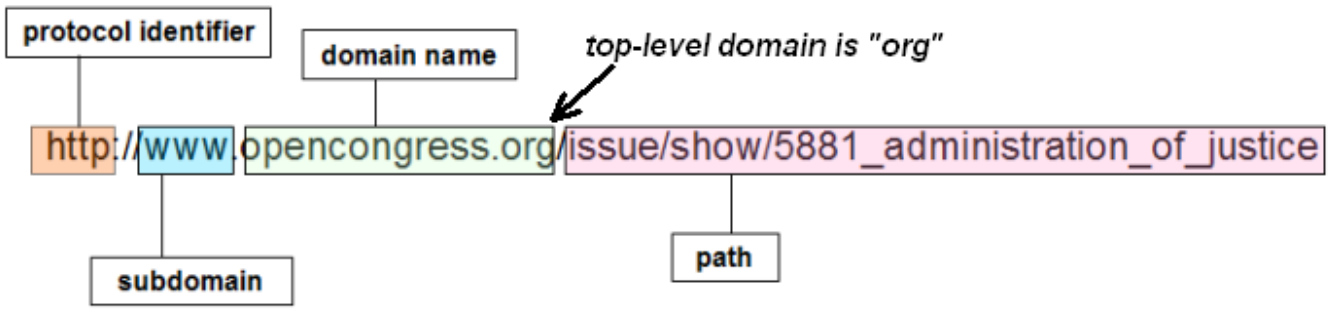
- This counteracts the problem of hidden nodes
- This is where a node can be heard by the WAP but not the node trying to transmit

Structure of the Internet

- The internet is a network of networks to allow computers to communicate globally
- The world wide web is a collection of webpages on computer connected to the internet
    - It uses the internet as a service to communicate the information contained within the pages

Uniform Resource Locator

- A URL is the address for a resource on the internet



Internet registries and registrars

- Registrars ensure that a domain name is only used one, and they hold records of all existing websites and details for domains that can be purchased
- They act as sellers of domain names

- Internet registries are global organisations with databases of all domain names currently issued
- They allocate IP addresses and track which domain names are associated to which IP address as part of the Domain Name System

Domain names and DNS

- A domain name identifies where a resource resides
- They are structured in a hierarchy
- Each domain name has one or more IP addresses

- The DNS matches the domain name to an IP
- When a webpage is requested using a URL, the browser requests the IP address from the DNS
    - If it is not found at that DNS, the search is move to a larger DNS database

Fully Qualified Domain Names

- A FQDN includes the host server name (www, mail, ftp) depending on the resource being requested

- websitename.co.uk is a domain name
- mail.websitename.co.uk is a FQDN

IP addresses

- An internet protocol address is a unique address that is assigned to a network device
- It indicates where a packet of data is being sent or is being sent from
- Routers use the IP address to direct the packet

Packet switching and routing

Packet switching

- It is a method of communicating packets of data across a network on which other communications are occurring simultaneously
- Cables are shared between many communications

Data packets

- Data that is being sent across a network is broken into smaller chunks called packets
- Packets are of variable size
- Packets contain a header and a payload
- They may have a trailer which contains a checksum or a Cyclical Redundancy Check
    - The CRC checksum is recalculated for each packet upon receipt and is used to check that no data changed during transmission
    - If the CRC differs, the packet is rejected and a new copy is requested

- The header includes the senders and recipient's IP addresses, the protocol and the number of the packet in the sequence
- Also include Time to Live or the hop limit
- Payload includes the information being sent

Routing packets across the internet

- Packet switching allows the packets to be sent along different routes
- They can be reassembled on the other end

Routers

- Routers are used to connect at least two networks
- Traversing between routers is called a hop
- The router reads the recipient's IP address in each packet and forward it to them along and fastest and least congested route
- Routers use a routing table to store and update the location of devices and find the fastest routes between them
- Dijkstra's algorithm is used to find the fastest route, but it is often a bottleneck

Internet security

Firewalls

- A firewall provides unauthorised access between two networks
- Typically consist of two NICs
    - One connected to the internal network and one to the external network
- Using firewall software, packets are analysed using packet filters or the firewall may act as a proxy server

Packet filtering

- Also known as static filtering
- The header of the packet is checked
    - This includes the destination and source IP
- If the destination IP is on the network's allowed list of IPs, it is accepted
- Static filtering can also block packets based on port numbers and protocols

Stateful inspection

- Also known as dynamic filtering
- It also examines the payload
- It creates temporary rules based on the rest of the conversation
- Routers keep conversation data in a Connection Table that is dynamically updated
- Can defend against port scanning since the Connection Table will not allow those connections as they were not requested by the computer

Proxy servers

- Intercepts all packets arriving and leaving a network

- Enables anonymous surfing
- The proxy caches commonly visited websites to return them immediately
    - Speeds up access to webpages and reduces traffic
- The proxy makes all requests on behalf of the user and caches them

Encryption

- Encryption is the process of turning plaintext to a ciphertext where the plaintext cannot be read without the key

Symmetric encryption

- Also known as private key encryption
- The same key is used to encrypt and decrypt data
- The key must be transferred through key exchange

Asymmetric encryption

- Uses a public and private key which are separate but related
- The public key is for others to encrypt data which they are sending to you which can then be decrypted with your private key
- Cannot deduce the private key from the public key

Digital signatures

- A digest of the unencrypted message is calculated
    - If there is a change in the message, the digest will be different
- In a conversation between A and B
    - The digest is encrypted using A's private key
    - This is the digital signature
        - Since only the owner of the private key could have encrypted it
    - The signature is added to the message adn then encrypted with B's public key
    - B then decrypts the message with their private key and decrypts the digest using the A's public key
    - The digest is then recalculated
        - If it matches, the message is deemed genuine (came from the sender and that the message did not change during transmission)
- The date and time of the message is recorded so that the message cannot be resent as it would change the value of the digest
- Can be used for any message

Digital certificates

- Hoax digital signatures can be created with a hoax private key to imitate a trusted individual
- Digital certificates are issues by Certificate Authorities who verify the trustworthiness of the sender of website
- The certificate allows the holder to use Public Key Infrastructure
- It contains:

    - a serial number
    - expiry date
    - the name of the holder
    - their public key
    - digital signature of the CA
- They operate within the Transport layer of TCP/IP using TLS.

Worms, Trojans and viruses

- These are all types of malware designed to cause inconvenience, loss or damage to programs, data or systems

Viruses

- A virus has the ability to self-replicate by spreading copies of itself
- Relies on the host files to be opened in order to spread
- Viruses become memory resident when the host program is run
- Programs become infected when they are loaded into memory

Worms

- Worms are a sub-class of viruses
- Do not need to be opened to spread
- Generally do not hide in other programs
    - Enter through a vulnerability
- A worm can spread to other computers
    - Methods such as emailing all members of the address book are used

Trojans

- A programs that appears to be something the user wanted
- When installed, the payload is released
- Often used to open a back door to the computer
- Cannot self-replicate
- Groups of infected computers are called bot nets

System vulnerabilities

- Malware exploits many different vulnerabilities to gain access to the system
    - Bugs in programs
    - Disabled firewalls
    - No anti-virus software
    - Social engineering
    - Phishing
- To avoid this programs should be checked for common bugs that allow greater access to a system such as buffer overflows

- To prevent social engineering people should be trained on how to physically protect access to systems and who it is safe to give details to
- To avoid phishing there should be a spam filtering system and more education in the use of caution
- There should be regular OS and anti-virus updates
    - New versions of OS will have less vulnerabilities that could be exploited and new variants of anti-virus software can detect new malware

TCP/IP

The TCP/IP protocol stack

- It is a set of protocols that working in four layers to pass incoming and outgoing data ip and down the layers

The application layer

- At the top of the stack
- It selects the appropriate high-level protocol for the application being used
    - Eg if the web browser is being used, the application layer would select HTTP(S)

The transport layer

- Uses TCP to establish an end-to-end connection with the recipient
- Data is split into packets
- The packet is labelled with the packet number and the total number of packets (packet sequencing)
- The port number is added
    - The port ensures it is handled by the correct application on the receiving end
- It also acknowledges received packets and requests lost packets

The network layer

- Adds the source and destination IP addresses
- Where routers operate
    - They use the IP addresses to forward the packets to the destination
- The IP and port is combined to form a **socket**

The link layer

- The physical connection between nodes
- Adds the MAC address to identify the NIC of the source and destination
- The MAC address changes at each hop
    - The source MAC address is of the device sending the packets for that hop
    - The destination MAC address is of the device receiving on that hop

What happens when packets are received?

- The MAC address is stripped by the link layer
    - Which then passes it to the network layer

- The IP addresses are removed by the network layer
    - Which passes it to the transport layer
- The transport layer determines which application to send the packet to in the application layer then removes the port number and reassembles the packets in the correct order
    - The packet is then sent to the application layer
- The application layer presents the data to the user

Fundamentals of databases

Modelling data requirements

- When designing a new database system, one of the first things that must be done is examine the data that needs to be input, processed and stored and determine what the data entities are

- An entity (record) is a category of object, person, event etc. about which data can be recorded
- Each entity in a database system has attributes (field)
- eg the patient entity may include the attributes of Title, Firstname, Surname, Address, Telephone

Entity descriptions

- Written in the format Entity1(Attribute1, Attribute2)
- Therefore the entity description for Dentist would be Dentist(Title, Firstname, Surname, Qualification)

Primary key

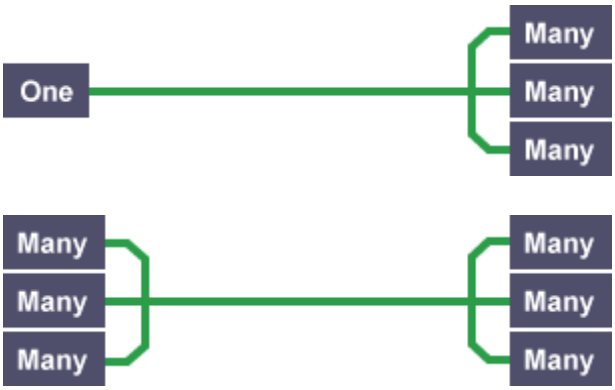
- The primary key uniquely identifies an entity
- In the dentist example none of the attributes are suitable
    - Therefore a numeric or string ID is used
- In the entity description, the primary key is underlined
- Dentist(DentistID, Title, Firstname, Surname, Qualification)

Relationships between entities

- One-to-one
    - Husband and wife, country and prime minister
- One-to-many
    - Mother and child, customer and order, borrower and library book
- Many-to-many
    - Student and course, stock item and supplier, film and actor

Entity relationship diagram

- A way to represent the relationships between entities



Relational database

- In a relational database, a separate table is created for each entity

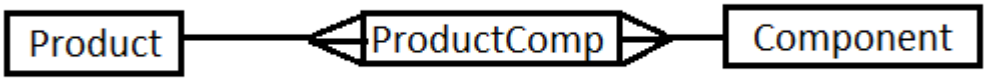
Foreign key

- A foreign key is an attribute that joins two tables
- It must be common to both tables
- The primary key of one table is the foreign key in the table it is linked to

Linking tables in a many-to-many relationship

- Two tables cannot be directly linked if there is a many-to-many relationship between them
- Eg. the relationship between Student and Course
    - A student takes many courses and the same course is taken by many students
- In this case, a link table is required

- In the general case this is how the link table works



- In the student example, the three tables will now have the attributes like this
    - Student(StudentID, Name, Address)
    - Enrolment(StudentID, CourseID)
    - Course(CourseID, Subject, ID)
- In this data model, the enrolment table has two foreign keys
    - The two foreign keys act as the primary key of the enrolment table.
    - This is called a composite primary key

Relational databases and normalisation

- A relational database is a collection of tables in which relationships are modelled by shared attributes

BookID	DeweyCode	Title	Author	DatePublished
88	121.9	Mary Berry Cooks	Berry,M	2014

BookID	DeweyCode	Title	Author	DatePublished
123	345.440	The Paying Guests	Waters,S	2014
300	345.440	Fragile Lies	Elliot,L	2015

- This table is described by Book(BookID, DeweyCode, Title, Author, DatePublished)

Normalisation

- Normalisation is a process to come up with the best design for a relational database

- No data is unnecessarily duplicated
- Data is consistent throughout the database
    - Since no data is duplicated, it should be consistent
- The structure of each table allows you to add any number of items
- The structure should allow complex queries relating data from different tables

First normal form

- A table is in first normal form if it contains no repeating attributes or groups of attributes

Second normal form

- A table is in second normal form if it is in first normal form and contains no partial dependencies
    - A partial dependency would mean that one or more of the attributes depends on only part of the primary key, which can only occur if the primary key is a composite key

Third normal form

- A table is in third normal form if it is in second normal form and contains no non-key dependencies
    - A non-key dependency is one where the value of an attribute is determined by the value of another attribute that is not part of the key

The importance of normalisation

Maintaining and modifying the database

- Data integrity is maintained since there is no unnecessary duplication of data
- If there is a change in data, it only has to be changed in one place

Faster sorting and searching

- Normalising will produce smaller tables with fewer fields
- This results in faster searching, sorting and indexing
- Since data is only held once it saves storage space

Deleting records

- A normalised database with correct relationships will not allow records in a table in the one side of a one-to-many relationships to be accidentally deleted

SQL

Keyword	Explanation
SELECT	Used to state which columns to query. Use * for all
FROM	Declares which table to select from
WHERE	Introduces a condition
INNER JOIN	Returns all rows where key record of one table is equal to key records of another
ORDER BY	The fields that the results are to be sorted by

Conditions

Symbol/Keyword	Explanation
=	Equal to
>	Greater than
<	Less than
<>	Not equal to
>=	Greater than or equal to
<=	Less than or equal to
IN	Equal to a value within a set of values
LIKE	Similar to
BETWEEN...AND	Within a range, including the two values which define the limits
IS NULL	Fields does not contain a value
AND	Both expressions must be true for the entire expression to be judged true
OR	If either or both of the expressions are true, the entire expression is judged true
NOT	Inverts truth

Data types

Data type	Description	Example
CHAR(n)	Character string of fixed length n	ProductCode CHAR(6)
VARCHAR(n)	Character string of variable length, max. n	Surname VARCHAR(25)
BOOLEAN	TRUE or FALSE	ReviewComplete BOOLEAN

Data type	Description	Example
INTEGER, INT	Integer	Quantity INTEGER
FLOAT	Number with a floating decimal point	Length FLOAT(10,2) (Maximum number of digits is 10 and maximum number after decimal point is 2)
DATE	Stores Day, Month, Year values	HireDate DATE
TIME	Stores Hour, Minute, Second values	RaceTime TIME
CURRENCY	Formats numbers in the currency used in your region	EntryFee CURRENCY

Altering a table structure

- The ALTER TABLE statement is used to add, delete or modify fields
- To add a column:

```
ALTER TABLE Employee
ADD Department VARCHAR(10)
```

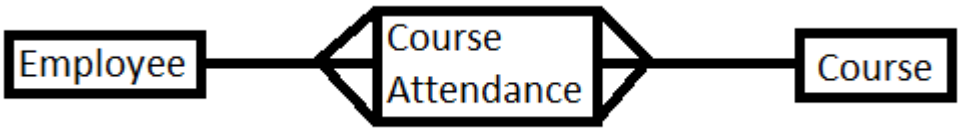
- To delete a column:

```
ALTER TABLE Employee
DROP COLUMN HireDate
```

- To change the data type of a column:

```
ALTER TABLE Employee
MODIFY COLUMN EmpName VARCHAR(30) NOT NULL
```

Defining linked tables



- The structure of employee table is:
  - EmpID - Integer (Primary key)
  - Name - 30 characters maximum
  - HireDate - Date

    - Salary - Currency
    - Department - 30 characters maximum
- The structure of the Course table is:
    - CourseID - 6 characters, fixed length (Primary key)
    - CourseTitle - 30 characters maximum (must be entered)
    - OnSite - Boolean
- The structure of the CourseAttendance table is:
    - CourseID - 6 characters, fixed length (foreign key)
    - EmpID - Integer (foreign key)
        - CourseID and EmpID form a composite primary key
    - CourseDate - Date

- The CourseAttendance table is created using the SQL statements:

```
CREATE TABLE CourseAttendance
(

  CourseID CHAR(6) NOT NULL,
  EmpID INTEGER NOT MULL,
  CourseDate DATE,
  FOREIGN KEY CourseID REFERENCES Course(CourseID),
  FOREIGN KEY EmpID REFERENCES Employee(EmpID),
  PRIMARY KEY (CourseID, EmpID)

)
```

Inserting, updating and deleting data using SQL

- INSERT INTO
- This statement is used to insert a new record into a table. The syntax is:

```
INSERT INTO tableName(column1, column2)
VALUES (value1, value2)
```

- UPDATE
- This statement is used to update a record. The syntax is:

```
UPDATE tableName
SET column1 = value1, column2 = value2,
WHERE columnX = value
```

- DELETE
- This statement is used to delete a record from a database table. The syntax is:

```
DELETE FROM tableName
WHERE columnX = value
```

Client-Server databases

- Many database management systems (DBMS) allow the DBMS server software to run on a server, and the client software runs on individual workstations
- The server processes requests for searches and reports
- Without client-server capability, databases that have to be accessed from many workstations would have to be copied to the workstation where client side software would process the search
- It takes a long tme to transmit mostly irrelevant data and the search may be longer as the client will be less powerful than the server

Advantages of Client-Server databases

- The consistency of the data base is maintained because only one copy of the data is held
- A powerful computer and a large database can be made available to many users
- Access right and security can be managed and controlled centrally
- Backup and recovery can be managed centrally

Problems with client-server

- When multiple users update a database at the same time, only one of the changes will be applied
- If User A accesses a record, which copies it to their memory and then changes the address for a customer
- User B then access the same record, and alters the credit limit and then saves the record and saves
- The database will store the original address with a different credit limit
- There are several method to avoid the loss of updates

Record Locks

- This prevents simultaneous access to objects in a database to prevent updates being lost or inconsistencies arising
- A record is locked whenever a user retrieves it or is editing it.
- Anyone else attempting to retrieve the same record is denied access until the action is completed.

Problem with record locking

- If two users are attempting to update two records, deadlock can occur where neither can proceed

```
User1                                   User2
Locks Customer A's record               Locks Customer B's record
Tries to access Customer B's record     Tries to access Customer A's record
Waits...                                Waits...
```

```
DEADLOCK!
```

Serialisation

- This ensures that transactions do not overlap in time and therefore cannot interfere with each other
- A transaction cannot start until the previous one finished

Timestamp ordering

- When a transaction starts, it is given a timestamp
- If two transactions affect the same object, the transaction with the earlier timestamp is applied first
- To ensure that transactions are not lost, every object in the database has a read timestamp and a write timestamp
    - These are updated whenever an object is read or written to
- At the start of the transaction, data is read causing the read timestamp to be set
- Before the updated data is written, the read timestamp is checked
- If it is not the same as the one set at the start, it will know that a transaction is taking place on the record

Commitment ordering

- This ensures that transactions are not lost when two or more users are simultaneously trying to access the same object
- Transactions are ordered in terms of their dependencies on each other and the time they were initiated
- It can prevent deadlock by blocking one request until another is completed

Regular languages

Mealy machines

Finite state machines

- Does not have output
- Sometimes called finite state automaton
- Model for computation

State transition diagrams

- Circles represent states that a system may be in
- Arrows represent the transition between states
- The start state has an arrow pointing into it
- Double circle is the accept state

 Finite state machine

State transition tables

- Alternative way to represent an FSM

 State transition table

Mealy Machines

- Is an FSM with an output
- Outputs determined by both its current state and the current input
- Like an FSM, it can be represented by a diagram or a state transition table

Applications of Mealy machines

- Can provide a mode of cipher machines
    - It can be designed to produce a cipher string when given a plaintext string
- Can also represent traffic lights, timers, vending machines and electronic circuits

Sets

Definition of a set

- A set is an unordered collection of values in which the value only occurs once
- A set can be defined in three ways

Defining a set by listing each member

- The members of the set are enclosed in curly brackets:
    - e.g. A = {2, 4, 6 ,8}

Common sets

- An empty set {} or ∅ has no elements
- The infinite set of natural numbers $\mathbb{N}$ = {0, 1, 2, ...}
    - Numbers used for counting
- Integers (+ve and -ve) Z = {-2, -1, 0, 1 , 2, ...}
- The set of all rational numbers is Q.
    - This includes all numbers that can be expressed as a fraction
- R is the set of all real numbers
    - Includes irrational numbers

Finite and infinite sets

- A finite set's elements can be counted by natural numbers
- The **cardinality** is the number of elements
- $\mathbb{N}$ is a countable infinite set
- R is not countable

Defining a set by set comprehension

- $ B = {n^2 \vert n \in \N \land n < 5 } $
- The | means such that

- The ∈ means element of
- The ^ means and
- Therefore B = {0, 1, 4, 9, 16}

Defining a set using compact representation

- $ A = { 0^n1^n \vert n } $
- This means that A is the set containing all strings with an equal number of 0s and 1s
- Therefore A = {01, 0011, 000111, 00001111}

Product of two sets

- The product of sets A and B, written A x B is the set of all ordered pairs (a,b), where a is a member of A and b is a member of B
- Eg A = {1,3,5} and B = {12,25,40}
- A x B = {[1,12], [1,25], [1,40], [3,12], [3,25], [3,40], [5,12], [5,25], [5,40]}

Subsets

- If every member of A is a member of B
    - A is then a subset of B
    - A ⊆ B
    - or B ⊇ A
- If A is a subset of B, but not equal to it, then A is a **proper subset** of B
    - A ⊂ B

Set operations

Union

- A ∪ B
    - This is a set containing every thing in A and B

Intersection

- A ∩ B
    - This set contains all members in common

Difference

- A \ B
    - All members that are in A but not in B

Regular expressions

- Used to:
    - match text patterns (searching for words in a word processor)
    - by compilers to check if the syntax is correct
    - validate user input

- A regular expression is used to specify a set of strings that satisfy given conditions
- The symbols:
    - **|** - separates alternatives
    - **?** - zero or one of the preceding element
    - ***** - zero or more of the preceding element
    - $^{+}$ - one or more of the preceding element

Regular language

- A regular language can be expressed by a regular expression
    - Or any language that a finite state machine will accept
- All finite languages are regular, since a regular expression can be created that includes all words in the language

The Turing machine

- Infinitely long tape divided into squares
- Read/write head
    - Reads symbols from the tape and makes decisions based on the contents of the cells and its current state
- Can be though of as an FSM with infinite memory
    - The FSM specifies the task to be performed
- The possible operations are:
    - Erase or write a symbol in the current cell
    - Move the read/write head left or right
- All Turing machines must have a halting state

Transition functions

- δ(Current State, Input symbol) = (Next State, Output symbol, Movement)
- δ(S1, 0) = (S2, 1, L)
    - This means if the machines is in State 1 and a 0 is read then write a 1, move to the left and change the state to State 2

The Universal Turing machine

- A Turing machine can theoretically represent any computation
    - Therefore a different turing machine is needed for every computation
- The Universal Turing machine can compute any computable sequence

- The machine **U** is an interpreter that reads a description of any Turing machine **M** then executes operations on data as **M** does. The description is written to the start of the tape, followed by the data **D**
- The definition of what is computable is anything that a Turing machine can compute
- The universal machine reads the description of the machine and the input from its own tape
    - This led to the stored program concept where the program and its data is held in memory

The Halting problem

- The Halting problem is about determining whether a program will halt with a given input
- Turing proved that a machine **H** to solve the Halting problem for all possible programs and their inputs cannot exist.
- The Halting problem shows that some problems are non-computable (cannot be solved by a computer)

Backus-Naur form

Defining the syntax of a language

- A meta language is used to describe another language
- Backus-Naur form is a meta language
- More efficient method for checking syntax of programs etc. than regular expressions
    - Constructs that could be described by regular expressions can be represented more compactly in BNF

BNF

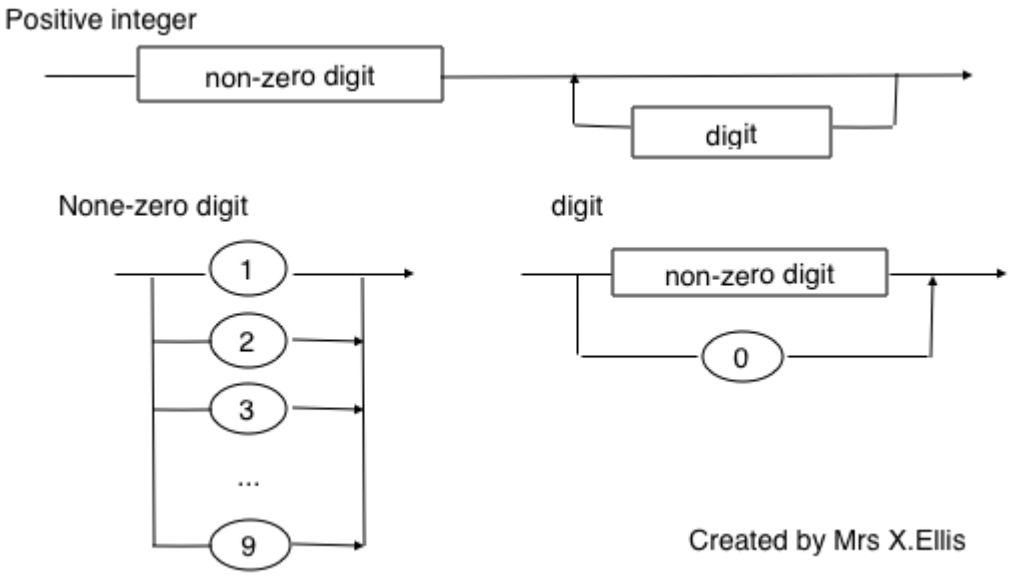
- LHS ::= RHS
- ::= means is defined by
- ::= is called a meta-symbol
- e.g. <point> ::= .
- <point> is called a meta-component
- | means or
- e.g. <digit> ::== 0|1|2|3|4|5|6|7|8|9

Recursion in a BNF definition

- The meta-component can be defined in terms of itself
- e.g. <variable list> ::= <variable><variable>, <variable list>

Syntax diagrams

- Ovals represent terminal elements (cannot be broken down further)
- Boxes are terminal elements (refer to another definition)



Reverse Polish Notation

- Reverse Polish, also known as postfix, is a method of writing expressions that is suited to use in computers

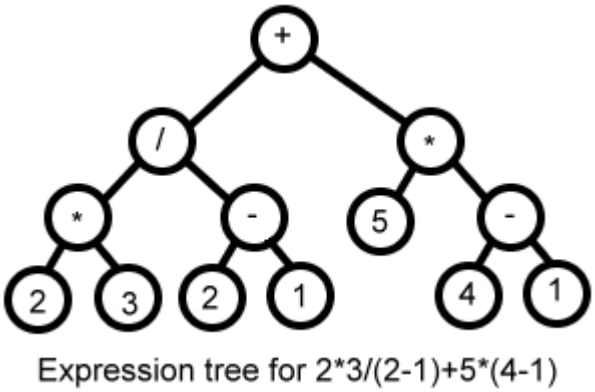
Advantages of RPN

- No need for brackets
- Produces expressions in a form suitable for evaluation on a stack
- Used in interpreters based on a stack

- The operator follows the operand

Binary expression trees

- An expression can be expressed as a tree
- An in-order traversal will give the infix expression
- A post-order traversal will give the postfix expression



Expression tree for 2*3/(2-1)+5*(4-1)

- The RPN will be 2 3 * 2 1 - / 5 4 1 - * +

Data structures

- An abstract data type is a description of how the data is viewed and the operation that can be performed
    - It is an example of data abstraction which created an encapsulation around the data and hides the details of the implementation
    - Called information hiding

Static and dynamic data structures

- A dynamic data structure is a portion of memory that can grow and shrink in size with the help of the heap
    - The heap is the area of memory where memory is allocated and de-allocated as required
- Dynamic data structures are useful in implementing structures such as queues where the maximum size of the data is not known in advance

- A static data structure is fixed in size such as an array
- The disadvantage of using an array to implement a dynamic data structure is that the size of the array has to be decided and now more memory can be added
- The memory that is allocated cannot be reallocated
- A static data structure is often simpler since pointers and other data does not need to be stored

Queues

- Queues are First In First Out (FIFO)
- New elements are added to the end and elements are retrieved from the front

Operations on a queue

- An abstract data structure is defined by structure and the operations that can be performed on it

- The following can be performed on a queue
    - EnQueue(item) // Add item to rear of queue
    - DeQueue(item) // Remove the front item
    - isEmpty() // Return true if empty
    - IsFull() // Return true if full

Implementing a linear queue

- Can be an array or a list

- As an item leaves the front of the queue, all other items have to move up one
    - This could have significant processing time on a large queue
- Can also be implemented with pointers to the front and rear with a variable holding the size of the array and the number of items in the queue
- However if many items are added and deleted from the queue, space is created at the front of the queue which cannot be filled, and items are added until the rear pointer points to the last element

Circular queue

- When the array is full and the rear pointer points to the last element of the array, it will be made to point at the first element

```
SUB initialise
  front := 0
  rear := -1
  size := 0
  maxSize := size of array
ENDSUB
```

```
SUB isEmpty
  IF size = 0 then
    RETURN True
  ELSE
    Return False
  ENDIF
ENDSUB
```

```
SUB isFull
  IF size = maxSize THEN
    RETURN True
  ELSE
    RETURN False
  ENDIF
ENDSUB
```

```
SUB enqueue(newItem)
  IF isFull THEN
    OUTPUT "Queue full"
  ELSE
    rear := (rear + 1) MOD maxSize
    q[rear] := newItem
    size := size + 1
  ENDIF
ENDSUB
```

```
SUB dequeue
  IF isEmpty THEN
    OUTPUT "Queue empty"
    item := NULL
  ELSE
    item := q[front]
    front := (front + 1) MOD maxSize
    size := size - 1
  ENDIF
```

```
  RETURN ITEM
ENDSUB
```

Priority queue

- Items are dequeued in the same manner
- Items with higher priority are at the front of the queue and items with low priority are at the back
    - Therefore, a new item can join at the front
- To implement this, the priority of an item is checked then starting at the rear it moves along until an item with the same or lower priority is found, at which point it is inserted

Lists

- Consists of items of the same type in which the same item can occur more than once
- The following operations can be performed on lists
    - isEmpty()
    - append(item) Adds item to end of list
    - remove(item) Removes the first occurrence of an item
    - search(item)
    - length()
    - index(item) Returns the position of the item
    - insert(pos, item)
    - pop() Removes and returns the last item
    - pop(pos) Removes and returns the item at the position pos
- An array can be used if the maximum number of data items is small and is known in advance
- The empty array must be declared in advance and could be used to hold a priority queue
- The algorithm for inserting an item into a sorted list

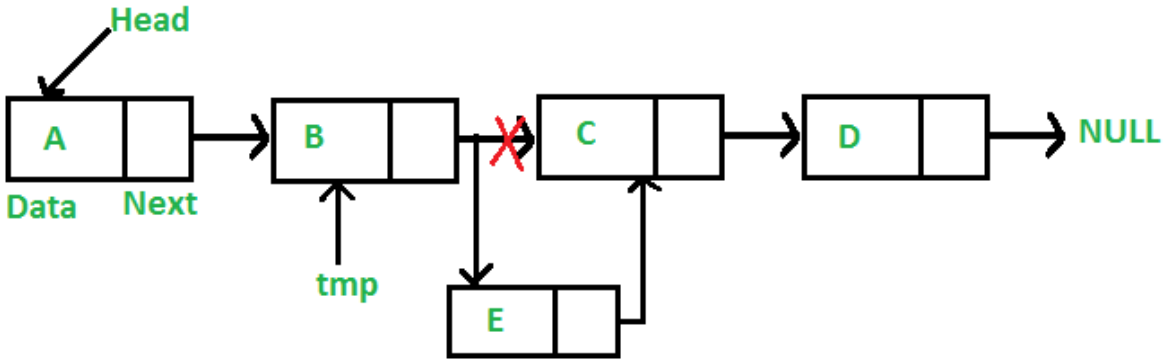
```
IF iSFull() THEN
  OUTPUT('FULL')
ELSE
  Find first greater
  starting at the end of the list, move all items one place
  Insert item
```

- To delete all items following the item deleted need to be moved one place
- This is done by copying all items to the previous index
- Then deleting the final item which is duplicated

Linked lists

- Dynamic lists are implemented as linked lists
- When items are added to the list, pointers are adjusted to point to the new memory locations
- When items are deleted the pointers are adjusted and the freed memory is de-allocated
- As nodes are added, memory locations are pulled from the heap

    - The heap is a pool of memory locations which can be allocated and de-allocated as required
- The pointers are then changed to maintain the correct sequence



Stacks

- Stacks are last in first out
- A stack can be implemented either as a static or dynamic data structure
- When implemented as a static structure in an array, two extra variables are required
    - Pointer to the top of the stack
    - Holding the size of the array (max size of stack)

Operations on a stack

- push(item) - adds item to TOP of stack
- peek() - returns top item from stack
- pop() - removes and returns item from TOP of stack
- isEmpty() - returns boolean
- isFull() - returns boolean

```
SUB isEmpty
  IF top = -1 THEN
    RETURN True
  ELSE
    RETURN False
  ENDIF
ENDSUB
```

```
SUB isFull
  IF top = maxSize THEN
    RETURN True
  ELSE
    RETURN False
  ENDIF
ENDSUB
```

```
SUB push(item)
  IF isFull THEN
    OUTPUT "Stack full"
  ELSE
    top := top + 1
    s(top) := item
  ENDIF
ENDSUB
```

```
SUB pop
  IF isEmpty THEN
    OUTPUT "Stack si empty"
  ELSE
    item := s(top)
    top := top - 1
    RETURN item
  ENDIF
ENDSUB
```

Overflow and underflow

- Stack has a maximum size
    - If an item is pushed to a full stack, it will overflow
- If the stack is empty
    - and a item is popped, it will underflow

Call stack

Holds return addresses

- This is the address of the instruction that control should return to when a subroutine finishes
- If subroutines are nested, the stack will grow as more return addresses are added which will be popped when the routine completes

Recursion

- With each call of a recursive routine a new return address is pushed to the stack
- When the recursion ends, the return addresses are popped one at a time each time the end of a subroutine is reached
- If the program is infinitely recursive, the stack will overflow

Holds parameters

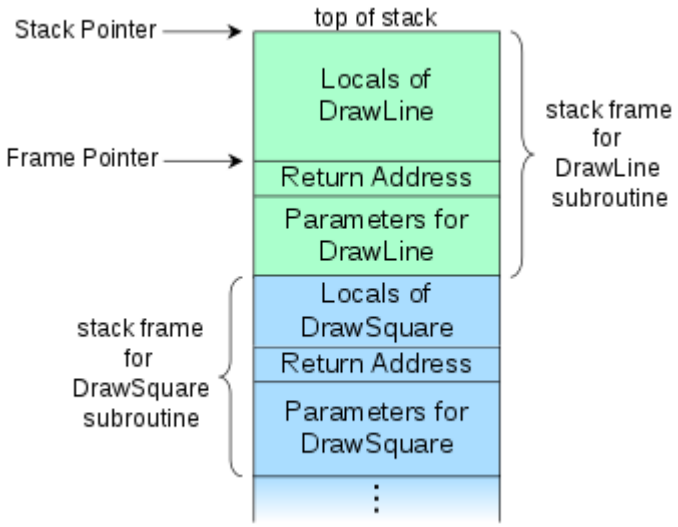
- Parameters required for a subroutine will be held on the call stack
- Each call of the subroutine will have a separate space on the call stack

Local variables

- Variables local to a subroutine is held on the call stack
- Storing local variables on the call stack is more efficient than dynamic memory allocation which uses the heap

Stack frames

- The call stack is made of stack frames
    - Each frame corresponds to a subroutine that is running



Hash tables and dictionaries

Hashing

- A hashing algorithm is applied to data of each record to produce an address
- A common algorithm is take take the data and mod it by the number of available addresses
    - `address = key mod availableAddresses`
- A synonym is where two keys hash to the same value
- A collision is where record keys hash to the same address
    - To deal with this place in the next available free space
- In a folding hash the item is divided into equal parts and added to give a hash then modded by the maximum number of addresses
- A string is hashed using the sum of the ASCII characters

Hash table

- A hash table is a collection of items stored so they can be quickly located
- The hash of the item to be inserted is calculated
- The item is then inserted at that address
    - If the space is taken, it is placed in the next available space

- How is a hash table searched for a value
    - Hash the value
    - Check the resulting cell
    - If item is there, return
    - If cell is empty the value is not present
    - If there is an item, keep moving along until it is found or a blank is found

Collision resolution

- As a hash table grows, there will be more collisions
- If the maximum number of data items is known, the table could be designed so that it is 70% full when all items are added to allow space for collisions

- Rehashing is how the next empty slot is found after a collision
- Following are some methods used:
    - Look linearly for next free slot
        - Loops to the start if end reached
    - Every n'th cell could be checked
    - The hash value could be incremented by a prime

Dictionaries

- Made up of pairs of keys and values
- When a key is supplied, the associated value is returned
- `IDs = {342: 'Harry', 634:'Jasmine'}`

Operations on dictionaries

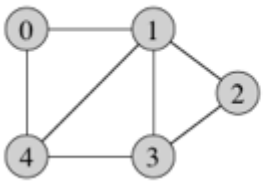
- Create a new dictionary
- Add new `key:value` pair
- Remove pair
- Change value of pair
- Return value associated with key
- Return True or False whether a key is present
- Return length of dictionary

Graphs

- A set of nodes connected by edges
- Where all edges are bidirectional, it is an undirected graph
- If all edges are one way, it is a directed graph
- The edges may be weighted

Implementing a graph

Adjacency matrix



	0	1	2	3	4
0	0	1	0	0	1
1	1	0	1	1	1
2	0	1	0	1	0
3	0	1	1	0	1
4	1	1	0	1	0

- In an undirected graph, the matrix will be symmetric
- If the graph was weighted, the weighting would be stored instead of 1/0

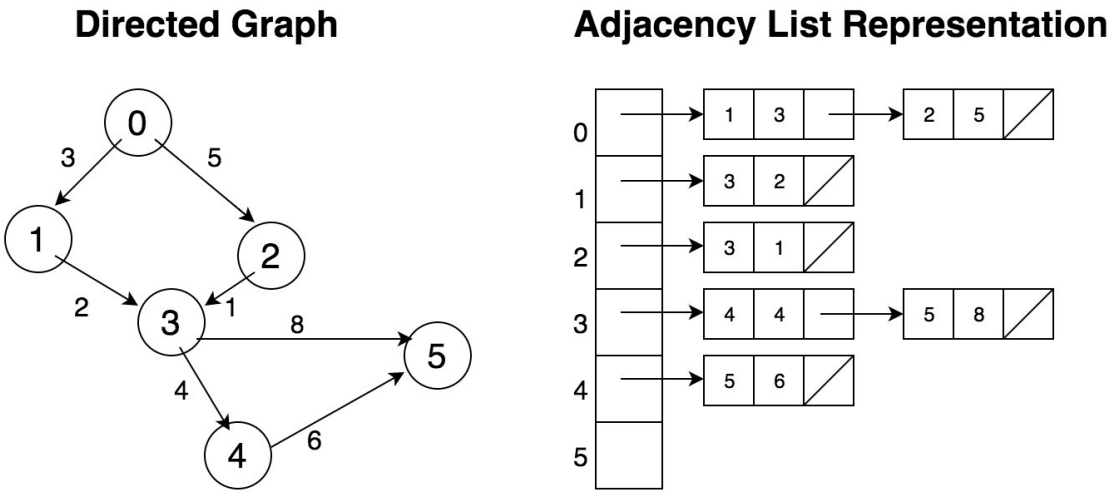
Advantages of an adjacency matrix

- Easy to add edge
- Easy to test for presence of edge

Disadvantages of an adjacency matrix

- In a graph with many nodes but few edges most cells will be empty
  - The larger the graph, the more memory is wasted
- It is hard to add or delete a node in a static two-dimensional array

Adjacency list



- More space-efficient
- A list of nodes is created
- Each node points to a list of all adjacent nodes
- Can be implemented as a list of dictionaries if there are edge weights

- Uses less memory when representing a sparsely connected graph

Applications of graphs

- Represent computer networks
    - Edge weights are the bandwidth
- Roads and towns
    - Edge weights are distances
- States in an FSM
- Web pages and links

Trees

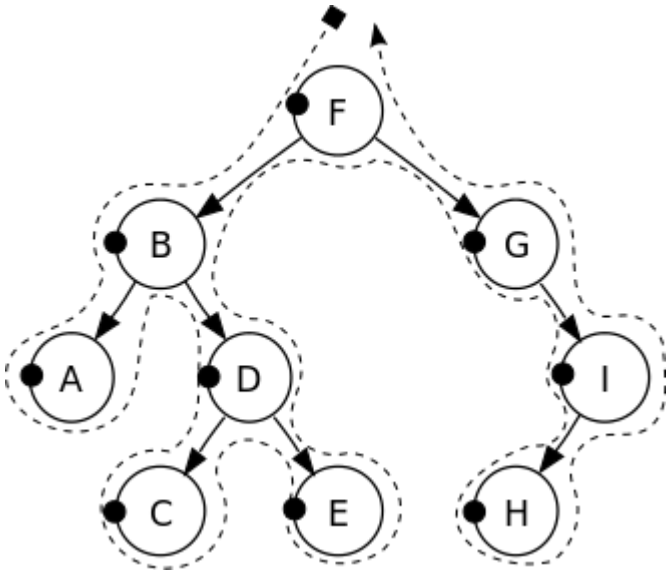
- **A tree is connected, undirected graph with no cycles**
- Connected: Always possible to find a route from a node to any other node
- No cycles: Cannot return to the same no without traversing an edge twice

Binary search tree

- Rooted tree with at most two children
- Holds items to allow them to be searched quickly
- New items can be added easily

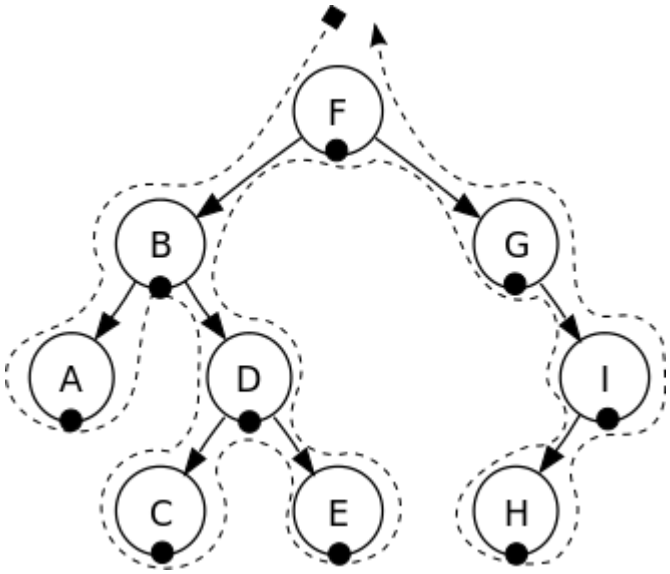
Traversing a binary tree

Pre-order



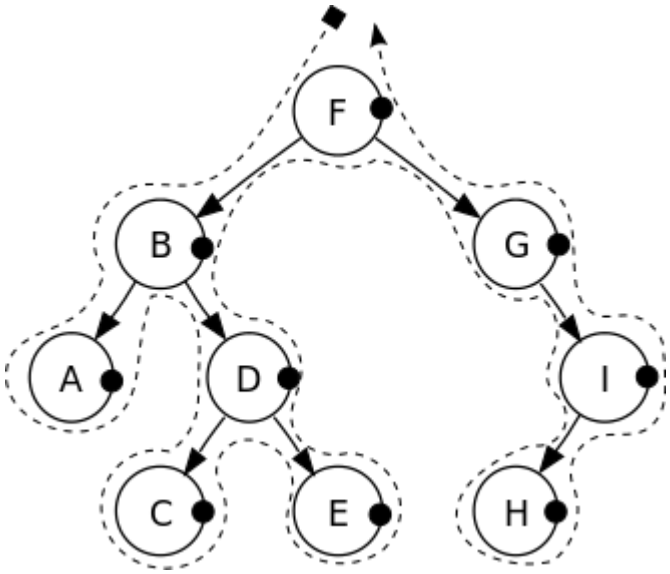
- Output node when first encountered
- F, B, A, D, C, E, G, I, H.
- Used to produce prefix notation, used in functional programming

In-order



- Output when passing underneath

Post-order



- Output when passing for the last time
- Used to produce RPN

Implementing a binary search tree

- Array of records with each node consisting of
  - Left pointer
  - Data item
  - Right pointer
- A pointer of -1 indicated there is no child on that side

tree[index]	Left	Data	Right
tree[0]	1	17	4
tree[1]	2	8	3

tree[index]	Left	Data	Right
tree[2]	-1	4	7
tree[3]	-1	12	6
tree[4]	5	22	8
tree[5]	-1	19	-1
tree[6]	-1	14	-1
tree[7]	-1	5	-1
tree[8]	9	30	-1
tree[9]	-1	25	-1

Vectors

- Vectors can be represented as a:
  - list of numbers
  - function
  - representation of a point in space

There are different notations

1. List [2.0, 3.14159, 2.71828]
2. A n-vector over $ \mathbb{R} $ can be written $\mathbb{{R}}^n$ 4-vector over R such as [2.0, 3.14159, 2.71828, -1.0] can be written $ \mathbb{R} ^ 4 $
3. can be shown as a function $ f : S \rightarrow \mathbb{R} $ $\text{ where } S \text{ is the set of } {0,1,2,3} \text{ and } \mathbb{R} \text{ is the set of all real numbers}$

- Example:
  - $0\rightarrow 2.0$
  - $1\rightarrow \pi$

Uses vectors

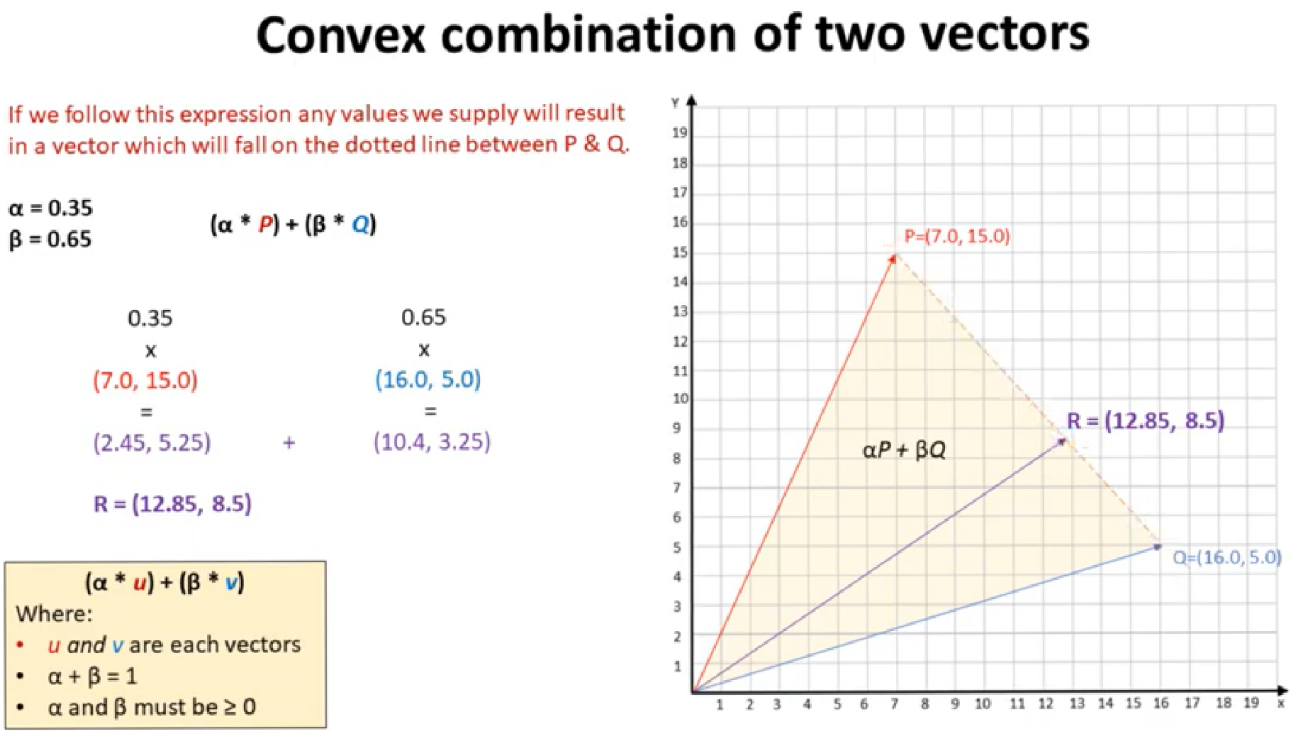
- Used when processing spacial information

1. When controlling a robot arm the coordinates need to be tracked so that the computer can work out how to move to the next location
2. Games and simulations have to work out how objects moving in 3D space can be represented
3. Planes need to take in account wind speeds and direction when setting and holding a course in 3D

Convex combination of vectors

- Used to find if object lies inside the area bound by two vectors

$$\alpha u + \beta v \text{ where } \alpha + \beta = 1 \text{ and where } \alpha , \beta \geq 0 $$



Dot product

$$ u • v = u_1v_1 + u_2v_2 + ... + u_nv_n$$

$$ [2,3,4] • [5,2,1] = 10 + 6 + 4 = 20 $$

- The dot product is a number, not a vector, and can be used to compare vectors

Finding angle between vectors

$$ cos \Theta = \frac{(u • v)}{(\Vert u \Vert • \Vert v \Vert)} $$

OOP and functional programming

Programming paradigms

- A paradigm is a style of a language
- **Procedural programming** rely on procedures defined by the user which are called in an order specified by the programmer
- **Object-oriented languages** make it possible to abstract details of the implementation in order to make code reusable and easy to maintain
- **Declarative languages** such as SQL is where you write statements to describe the problem to be solved and the language decides how to implement it
- **Functional programming** is where functions are used as the building block of the program. Statements are written as a series of functions which accept data and return an output

Object-oriented programming

- An object-oriented program is a number of interacting objects each of which is responsible for their own data and operations on that data
- The program code creates objects and allows them to communicate by sending messages and receiving answers

Object attributes and behaviours

- Object will have attributes
  - For example a person object would have the attributes first name, last name and data of birth
- An object has state
  - For example a radio can be on or off, tuned to a station or set to a volume
- The object's behaviours are the actions it can perform

Classes

- A class is a template for an object where the attributes and methods are defined for all object of that class

```
ClassName = Class
  Public
    //define all methods
  Private
    //define instance variables
End
```

- Generally, instance variables are private and methods are public
- This is so that other classes may use the methods belonging to another class but not see or change their attributes

- Information hiding it important
  - This is where classes cannot directly access the private attributes of another class

Instantiation

- This is how objects are created
- If you wanted to add a book to a stock system
- All objects in a class have the same structure and methods but each have their own data

```
book1 = new StockItem("PT123", "Book" , "Computer Science", 35)
```

- This creates a reference variable called book of which is an instance of StockItem with the defined attributes

- Reference variables are named areas in memory where information can be stored

    - It does not store the object but instead holds a pointer

- In variable reference diagrams the reference variable is shown as circles and primitive data types are shown as rectangles

Sending messages

- Messages are either getters or setters
- Getters are functions and setters are procedures

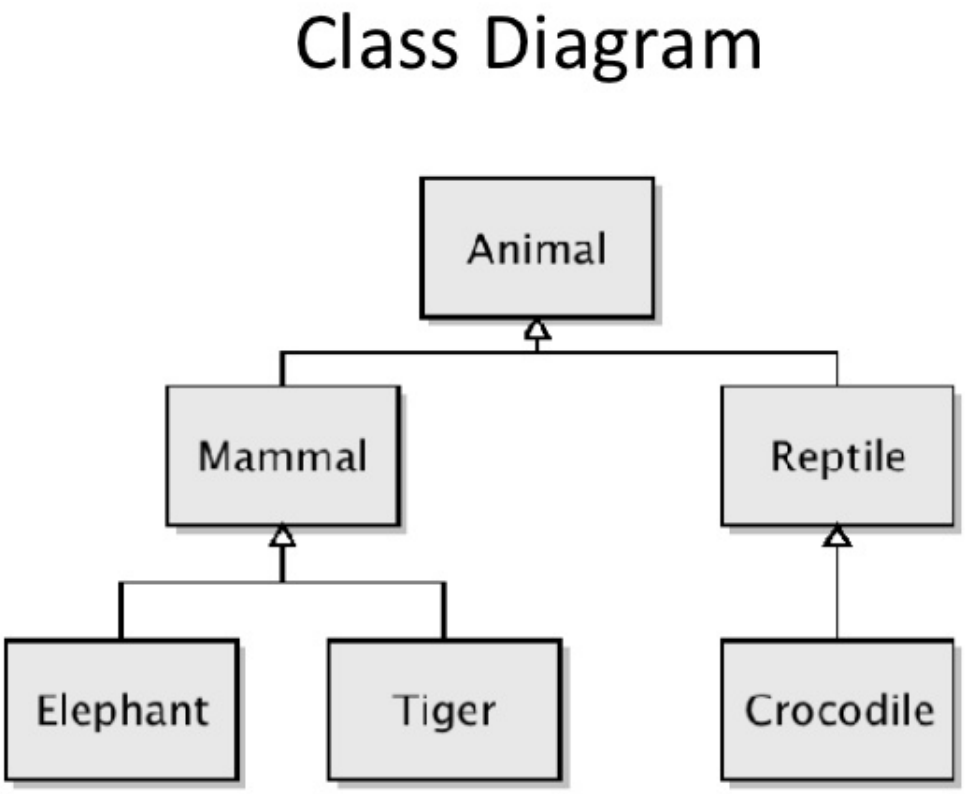
- The state of an object can be examined or changed by sending it a message

Encapsulation

- An object encapsulates its state and methods
- The data and methods are wrapped into a single entity so that the attributes and methods of one object cannot affect another object
- Encapsulation allows programmers to work on classes and not worry about how other it may affect other parts of the system
- Methods from other classes can be used without knowing how they work
- This is related to information hiding, where details of an object are hidden so that messages must be used to interact

Inheritance

- A class can inherit data and methods from a parent class
- A child class is called a subclass and a parent class is called a superclass



- Inheritance is used when object A is an object B
- Eg. Cat is an Animal and therefore can inherit from animal

- `Cat = Subclass(Animal)` or `Class Cat(Animal)` or `public class Cat extends Animal`

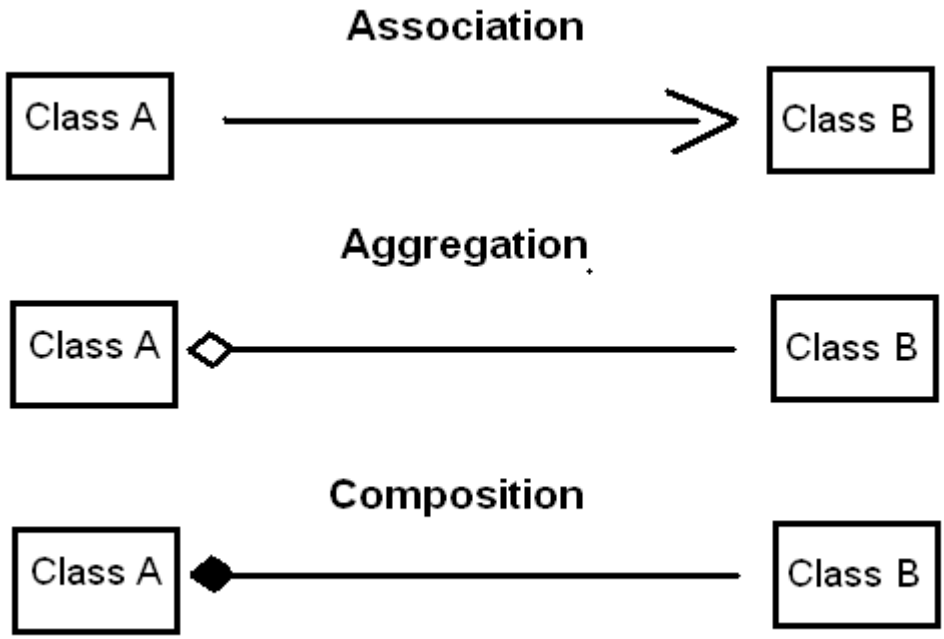
OOP Design principals

Association, aggregation and composition

- Association is described as a "has a " relationship
  - Eg. a teacher has a student and a student has a teacher
- There is no ownership
  - They have their own life-cycle and can be created and deleted independently

- Aggregation is a type of association
- Occurs when a class is a collection for other classes, but the contained classes do not have a strong life-cycle dependency on the container

- Composition is a strong form of association
- If the container is destroyed, all instances are destroyed



Polymorphism

- This is the ability to process object differently depending on the class
- For example when a move command is sent to a Cat class, it may move two spaces whilst a Rodent will only move one
- The contents of the method is different but the name is the same
- This is called overriding

```
Animal = Class
  Public
    Procedure moveLeft
    Procedure moveRight
  Protected
    Position : Integer
  End
Cat = Subclass(Animal)
  Public
    Procedure moveLeft (Override)
    Procedure moveRight (Override)
    Procedure pounce
  Private
    Name : string
  End
```

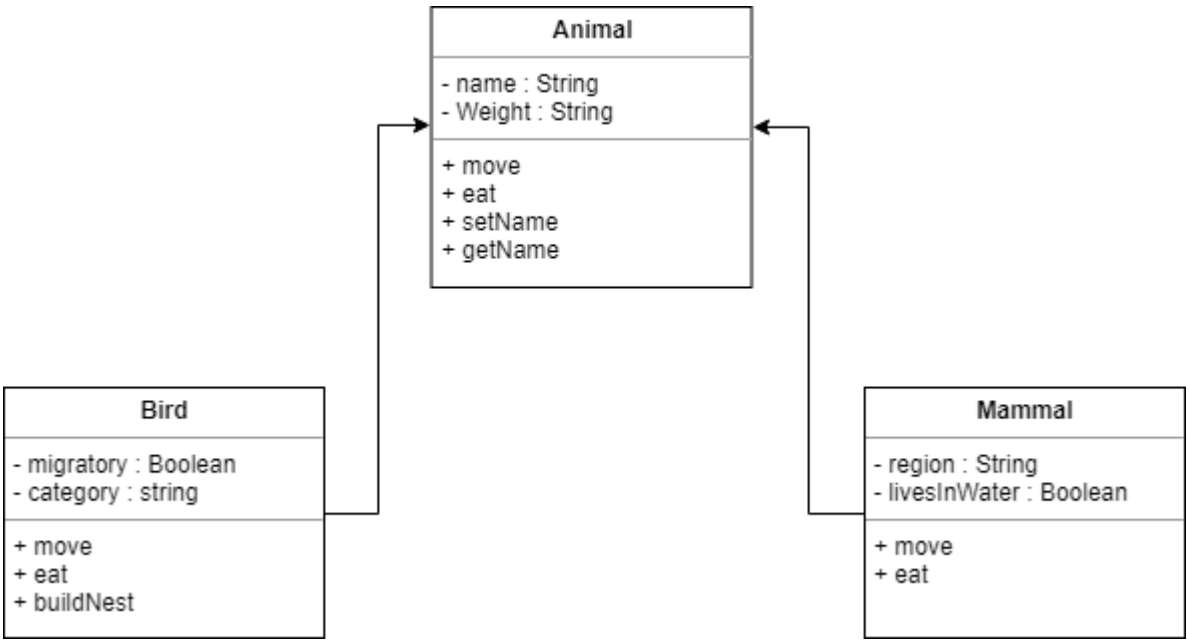
Access modifiers

- If a method or instance variable is declared
  - Private only code within the class can access it
  - Public code within any class an access it
  - Protected

- Restricts access to members of a subclass or members in the same package or library of classes

Member is accessible	Public	Protected	Private
Within the defining class	Yes	Yes	Yes
Via inheritance	Yes	Yes	No
Via a reference to an object of the class	Yes	Only if it is in a subclass or if it is in the same package	No

- In class diagram, private is - and public is +



Favour composition over inheritance

- Composition allows greater flexibility
  - It is less rigid in relationships between objects
- An object may consist of many other objects but cannot be said to inherit their characteristics
- Eg. A house class will be made up of Walls, Doors, Windows and Roof classes

```
House = Class
  Public
    Procedure drawHouse
    Procedure getHousePosition
    Procedure setHousePosition
  Private
    TheWall : Wall
    TheDoor : Door
    WindowLeft : Window
    WindowRight : Window
    TheRoof : Roof
  End
  Wall = Class
```

```
    Public
      Procedure drawWall
    Private
      WallHeight : Real
      WallWidth : Real
      WallColour : Integer
```

Programming to an interface

- Sometimes a number of different classes all need to understand a particular set of messages
- For example switchOn could be sent to a microwave, lamp, oven and watch
    - They will respond differently depending on the class
- An interface is a collection of abstract methods that a group of unrelated classes may implement
- The methods are specified in the interface but implemented in the class using the interface
- The programmer does not need to know how the objects in each class will respond to the message

```
Public interface Switches
  Procedure SwitchOn
  Procedure SwitchOff
  Procedure SetTimer(aTime)
  Procedure GetTimer
End
```

- A programmer who wants to implement the Switches interface in the Microwave class would do `Class Microwave Implements Switches`
- The objects must be able to handle any of the messages from the interface
- An advantage is new classes that use the interface can be added without affecting existing classes

Encapsulate what varies

- Used to reduce maintenance and testing
- If a change needs to be made only the mode containing the concept should have to change
- The modules that are most likely to change should be considered and encapsulated during the design stage
    - If they need to be changed in the future, the amount of code that needs to be modified is minimised
- The concept could be implemented using interfaces where the methods are implemented differently by each class

Advantages of OOP

1. Forces designers to plan extensively, which leads to better designs with fewer weaknesses
2. Encapsulation allows code for an object to be written, tested and maintained independently of other objects
3. Knowledge of how methods are implemented is not required once an object has been created

4. New objects with small differences to existing ones can be easily created
5. Objects that have already been defined, coded and tested can be easily re-used in different programs
6. Easier to maintain due to the modular structure

Functional programming

What is a function?

- A mapping of a domain to a co-domain
    - Eg $f: A \rightarrow B \vert f(x) = x^2$
- Does not have to be algebraic

Defining a function

- Define a function as follows: `add3int x y z = x + y + z`
- Giving a function an input is called function application
    - `add3int 1 2 3`

First-class objects

- A first-class object is an object which may:
    - appear in expressions
    - be assigned to a variable
    - be assigned as an argument
    - be returned in a function call
- Functions are first-class objects and therefore can be passed as arguments

Advantages of functional languages

Statelessness

- The value of a variable cannot change
    - The variable is said to be immutable
    - This makes the program stateless

There are no side effects

- Functions only calculate and return and therefore have no side effects
- Since the value of a variable cannot be changed, if a function is called twice with the same parameters, it will return the same result

Less bugs

- It is easier to write bug free programs
- Small functions that can be easily be proved correct are used to construct larger, more complex functions

Functional composition

- Two functions can be combined where the result of one function becomes the input of another
    - g(f(x))
        - Here the result of f with the input x is sent to g

To produce g(f(x)) with x = 4 in Haskell

```
f x = x + 3
g x = 2 * x^2
(g.f) 4
```

Types and type-classes

- In Haskell, types are sets of values, and type-classes are sets of types
- Integer, Int, Double, Float are all in type-classes Num.
    - Integer can represent any value whilst Int in bounded
- The type of a function should be declared explicitly

```
sumOfSquares :: Integer -> Integer -> Integer
sumOfSquares x y = x^2 + y^2
```

The last type in the type declaration is the type that is returned

- Type variables represent any type

Function application

Higher-order functions

- These take a function as an argument or returns a function as a result

How are functions evaluated?

All functions in Haskell only take one argument Then how are statements such as

```
add3Int x y z = x + y + z
```

possible

- A function takes a parameter at a time

```
add3Int :: integer -> integer -> integer -> integer
-- The type declaration can also be written as
add3Int :: integer -> (integer -> (integer -> integer))
```

What happens when `add3Int 2 4 5` is called

- The function is applied to the arguments
- It takes the first argument, 2, and produces a function to add 2 to its arguments
- This function then produces another function that adds 5 to the value of the previous function

- The function add3Int takes the argument of 2 and returns a function of type `(integer -> (integer -> integer))`
- This function takes the argument of 4 and returns a function of type `(integer -> integer)`
- This function takes the argument of 5 an returns an integer

Partial application

- This uses the way functions are evaluated by decomposing multi-argument functions into smaller functions with fewer arguments

```
add :: Integer -> Integer -> Integer
add x y = x + y
```

- This means (add 3) 4
- This function can then be partially applied as an argument for another function

```
addSix :: Integer -> Integer
addSix = add 6
```

- The function add required more that one argument and only one is passed
- It then returns a new function that takes the remaining argument and returns the result

- Partial application means fixing the values of inputs to a function to produce a more specific function

Map

- Map is a high-order function that takes a list and the function to be applied to all elements and returns a list after applying the function

- Eg:

```
map (max 3) [1,2,3,4,5]
```

- This will return `[3,3,4,5]`

Filter

- A high-order function which takes a predicate and a list and returns the items that satisfy the condition

- Eg:

```
filter (>6) [2,5,6,8,9]
```

- This will return `8,9`
- You can write your own predicate
- Eg:

```
isEven n = n `mod` 2 == 0
filter(isEven) [1,2,3,4,5,6]
```

- This will return `[2,4,6]`

Fold

- Reduces a list to a single value using recursion

```
foldl (+) 0 [2,3,4,5]
```

- The calculation that takes place is (((0 + 2) + 3) + 4) + 5
- foldl starts the recursion at the left, foldr starts at the right

Lists in functional programming

- Like variables, lists are immutable

```
names = ["Anna", "Bob", "Jo", "Keira", "Tom", "George"]
numbers = [3,7,14,83,2,77]
head names
tail names
head numbers
tail numbers
```

- This will return

```
["Anna"]
["Bob", "Jo", "Keira", "Tom", "George"]
[3]
[7,14,83,2,77]
```

- Tail can be applied repeatedly

- The following defines a new list and checks if it is empty

```
let newList = []
null newList
```

- This will return `True`
- To prepend an element to a list use `:`
    - Eg. 5:Numbers
- To prepend a list use `++`
    - `[6,10] ++ Numbers`
    - `Numbers ++ [6,10]` will append
- `length numbers` returns the length of numbers