Computing COMP3

Programming Paradigms

- **Programming Paradigms**
  - **Imperative (eg. Pascal)**
    - Statements that perform actions
    - Executed in order
    - Manipulate variables and data structures
  - **Functional (eg. Haskell)**
    - Manipulate lists not variables
    - Define mathematical formulas that manipulate lists
    - Solutions consist of a series of function calls
  - **Logic (eg. Prolog)**
    - Facts and rules are used to built up a knowledge base
    - An inference engine use the knowledge base to answer queries presented to it
  - **Event-driven (eg. VB.NET)**
    - Event is an action or occurrence detected by the program eg. the user pressing a button
    - Procedures are executed in response to actions
  - **Object-oriented (eg. Java)**
    - Uses classes and objects to split up programs
- **Object-Oriented Programming**
  - **Class**
    - Collection of objects that share the same properties
    - Have the same data fields and methods
  - A **Class Definition** is pattern or template that can be used to create objects of that class
  - **Object**
    - An instance of a class
    - Has a collection of data fields and methods
  - **Instantiation** - When an object is defined based on a class
  - **Encapsulation**
    - Hide the internal methods and variables
    - Methods have to be access via an interface
  - **Inheritance**
    - Relationship between classes, one shares its structure with another
    - The child class can use the parent's methods and variables as well as it own
  - **Polymorphism**
    - Changing the implementation of inherited methods it make them more appropriate to itself
- **Recursion** This is when a procedure calls itself
- A recursive procedure must have
  - **General Case**
    - Returns the result of a call to itself with an argument of n
  - **Base Case**

        - Returns a value that does not rely on a recursive call
- **Stack frame** this is the locations in the stack that store values referring to a call to a procedure
- When a procedure is called its return address is stored on the stack
- When a procedure ends its stack frame is popped
- **Abstract Data Type** a data type whose properties are specified independent of any programming language
- **Static data structure** is a structure in which the space in memory is defined before run time
- **Dynamic data structure** is a structure in which the memory taken up by the structure can change during run time
- **Dynamic Allocation** memory space is only allocated when it is needed
- The **Heap** is an area in memory that is available for application programs for dynamic allocation
- **Pointer** a variable that holds the memory address of another variable
- **Memory Leakage** occurs when memory locations are not released after they are finished being used and no memory is left to be allocated
- **Lists**
    - A collection of elements with an inherent order
    - **Linear List**
        - Data is stored in adjacent memory locations
        - Maximum size defined at design time
        - **Insertion** - Store the element at the next free space, increment `nextFree` variable
        - **Deletion** - Shift elements after the one to deleted one forward and decrement `nextFree` variable
    - **Linked List**
        - Data is stored wherever in memory but with a pointer to the next variable in the list
        - The final element in the list has a **null pointer**
        - Two lists, one storing values the other of free space
        - Two pointers, one pointing to the front of each list
        - **Insertion**
            - `Check if NextFree is NULL`
            - `Set a temporary variable to the value of NextFree node's pointer`
            - `Set NextFree node's value to data to be stored`
            - `Set NextFree node's pointer to value of previous node's pointer`
            - `Set previous node's pointer to NextFree`
            - `Set NextFree to the value of the temporary variable`
        - **Deletion**
            - `Check if Start is NULL`
            - `Set previous node's pointer to the value of the node to be deleted's pointer`

            - Set the deleted node's pointer to the value of NextFree
            - Set NextFree to the address of the deleted node
        - **Search**
            - Step through the list from the start, following pointers
- **Stacks**
    - Stacks are **LiFo** (Last in, First out)
    - **Push** adding an item to the top of a stack
    - **Pop** removing an item from the top of a stack
    - **Linear list implementation**
        - Use an array to store the data
        - Have a pointer that points to the top of the stack
        - **Push**
            - Check if stack is full
            - Increment TopofStack pointer
            - Store data in element pointed to by TopofStack pointer
        - **Pop**
            - Check if stack is empty
            - Return value held element pointed to by TopofStack pointer
            - Decrement TopofStack pointer
    - **Linked List implementation**
        - Uses a linked list to store the stack
        - Element are added and removed from the start of the list
        - **Push**
            - Set a temporary pointer to the value of NextFree node's pointer
            - Set NextFree node's value to the data
            - Set NextFree node's pointer to Start
            - Set Start to NextFree
            - Set NextFree to the value of the temporary pointer
        - **Pop**
            - Set a temporary pointer to the value of Start node's pointer
            - Return the Start node's value
            - Set Start node's pointer to NextFree
            - Set NextFree to Start
            - Set Start to the value of the temporary pointer
- **Queue**
    - Is a **FiFo** (First in, First out) data structure
    - **Shuffle Queue**
        - Stored as a linear list

        - Two pointers, pointing to the front and rear of the list
        - New items added to the back of the queue, rear pointer incremented
        - Items removed by shuffling all others down and decrementing rear pointer
        - Usually not a realistic option
    - **Circular Queue**
        - Stored in a linear list
        - Two pointers, pointing to the front and rear of the queue and a variable holding the size of the queue
        - Queue wraps around to the beginning of the array
        - Difficult to differentiate between a full and empty queue, so a variable holding the queue size is also stored
        - **Add Item**
            - `Check if Queue is full`
            - `Increment rear pointer`
            - `If rearPointer greater than the array's size set rearPointer to 1`
            - `Store the data at the element pointed to by rearPointer`
        - **Remove Item**
            - `Check if Queue is empty`
            - `Return the item pointed to by the frontPointer`
            - `Increment frontPointer`
            - `If frontPointer greater than the array's size set frontPointer to 1`
    - **Linear Queue**
        - Stored in a linked list
        - Two pointer, pointing to the front and the rear of the list
        - **Add Item**
            - `Set a temporary pointer to the value of NextFree node's pointer`
            - `Set NextFree node's value to the data`
            - `Set NextFree node's pointer to NULL`
            - `Set RearPointer Node's pointer to Nextfree`
            - `Set RearPointer to NextFree`
            - `Set NextFree to the value of the temporary pointer`
        - **Remove Item**
            - `Set a temporary pointer to the value of FrontPointer node's pointer`
            - `Return the FrontPointer node's value`
            - `Set FrontPointer node's pointer to NextFree`
            - `Set NextFree to FrontPointer`
            - `Set FrontPointer to the value of the temporary pointer`
    - **Priority Queue**

        - All items have a priority as well as a value
        - When an element is taken from the queue the element with the highest priority is chosen
        - Can be implemented using a linked list, items are inserted into order based on their priority
        - Can be implemented using a binary tree
- **Graphs**
    - **Graph** - Made up of vertices joined by edges
    - **Labelled Graph** - A graph in which the edges are labelled or given a value called a weight
    - **Unlabelled Graph** - A graph without weighted edges
    - **Vertex** - A node, a point on the graph
    - **Edge** or **Arc** - A line that connects two vertices
    - **Digraph** - A graph in which its edges have a direction
    - **Closed Path (Circuit)** - A sequence of consecutive edges that start and end at the same vertex
    - **Cycle** - A closed path in which all vertices and edges are different
    - **Adjacency Matrix** - Use when the graph has many edges
    - **Adjacency List** - Use when there are many vertices but few edges
    - **Simple Graph** - An undirected graph without multiple edges and in which each edge connects two different graphs
    - **Tree** - A connected undirected graph with no cycles
    - **Rooted Tree** - A tree in which one vertex has been designated as the root and every edge is directed away from the root
    - **Traveller's Problem** - A route that visits each vertex exactly once before returning to the starting point
    - **Explorer's Problem** - A route that traverses each edge exactly once before returning to the starting point

<u>Standard Algorithms</u>

- **Insertion Sort**
    - Algorithm

```
for (int i = 2; i <= NumOfItems; i++)
{
      item[0] = item[i];
      ptr = i - 1;
      while (item[ptr] > item[0])
      {
            item[ptr + 1] = item[ptr];
            ptr--;
      }
      item[ptr + 1] = item[0];
}
```

    - In words
        - The second value is stored in a temporary variable
        - All values to the left of the second value that are larger than it are shifted right

        - The second value is put into the new gap
        - This repeated with the third, fourth, etc. values
- **Binary Search**
    - Algorithm

```
int BinarySearch(int list[], int item, int first,
                                          int last)
{
      if (first > last)
      { return NOT_FOUND;}
      else {
            int mid = (first + last) div 2;
            if (list[mid] > item)
            {
                  return BinarySearch(list[], item, first,
                                                      mid);
            } else if (list[mid] < item)
            {
                  return BinarySearch(list[], item, mid,
                                                   last);
            } else
            {
                  return mid;
            }
      }
}
```

    - In words
        - Find midpoint in list
        - Test is the value at the midpoint is the correct one
        - If the item at the midpoint is larger than the one to be found take the smaller list if not take the larger on
        - Repeat with the smaller list
- **Binary Search Tree**
    - A tree in which each node has up to two children
    - The left child is smaller than the node, the right larger
    - Stored as three parallel arrays, one with the value of the node, the other two holding pointers to the left and right child nodes
    - Three types of traversal algorithms
        - Pre-order
            - Output the root
            - Search the left
            - Search the right
        - Inorder
            - Search the left
            - Output the root
            - Search the right
        - Post-order

        - Search the left
        - Search the right
        - Output the root
- **Hashing**
    - **Hash key** - This is the kay that the hash function is applied to
    - **Hashing** - The process of applying a hash function to a key to generate a hash value
    - **Hash value** - The value generated by the application of the hash function to the key.
    - **Hash function** - A function H, applied to a key k, which generates a hash value H(k) of range smaller than the domain of k
    - **Hash table** - The table that data should be stored in, the index of the data is the hash value
    - **Collision** - A collision occurs when two or more different keys hash to the same hash value
    - **Open hashing** - Method in which a collision is resolved by storing the record in the next available location
    - **Closed hashing** - Method in which a collision is resolved by appending the value onto a linked list in the correct location
    - **Rehashing**
        - The key is hashed again because the initial hash resulted in a collision
        - **Linear Rehash** - The original hash is incremented by 1 mod N, then 2 mod N until an empty slot is found, in a table of size N rows
- **Simulations**
    - Can represent real and imaginary situations
    - Simulations allow users to study or try things that would be difficult or impossible to do in real life
    - **Model** - An abstraction of reality
    - **Simulation** - The imitation of a real process of a real life system
    - **Entity** - The components that make up a system
    - **Attributes** - The characteristics of the entities
    - **State History** - A chronological set of descriptions at each of the state instants
    - **State** - The state of a system at any instant is determined by where the entities are, what they are doing and their attributes
    - **Event** - The result of a change in state
    - **Activity** - A time-consuming process

<u>Operating Systems</u>

- Role of an operating system
    - Hides the complexities of the hardware from the user
    - Manage the hardware resources, allocate processors, memory and I/O devices
    - Manage data storage
- **Virtual Machine** The apparent machine that the operating system presents to the user, hiding the complexity of the computer's hardware behind operating system software

- **System Program** - A program that manages the operation of a computer
- Programs that manage resources
    - **Processor scheduling** manages the use of the **processor**
    - **Memory management** manages the use of **memory**
    - **I/O management** manages the use of **input/output devices**
    - **File management** manages the **data**
- **APIs (Application Programming Interface)**
    - A layer of software that allows application programs to call on the services of the operating system
- **Types of Operating System**
    - **Interactive**
        - The user and the computer are in direct two-way communication
        - User supplies commands
        - Computer displays the results of commands
    - **Real Time**
        - Inputs are processed in a timely manner so that the output can affect the source of the inputs
        - Have four requirements
            - Have to support application programs which are non-sequential, those without a START-PROCESS-END structure
            - Deal with a number of events which happen in parallel and at unpredictable times
            - Carry out processing and produce a response in a specific time interval
            - Some are safety-critical, so they must be fail-safe and guarantee a response within a specified time interval
    - **Network**
        - A layer of software added to the operating system of a computer connected to a network
        - Intercepts commands that reference resources elsewhere on the network e.g. a file server, then redirects the request to the resource in a manner completely transparent to the user
    - **Device**
        - **Sandbox** - A tightly controlled set of resources for guest programs to run in
        - Advantages of having an OS on a device
            - Device can multitask
            - Can operate in real time
            - Hardware can be changed without having to change the application code
            - New applications can easily be added
            - Basic functionality can be changed by upgrading the operating system instead of scrapping the hardware
    - **Embedded**

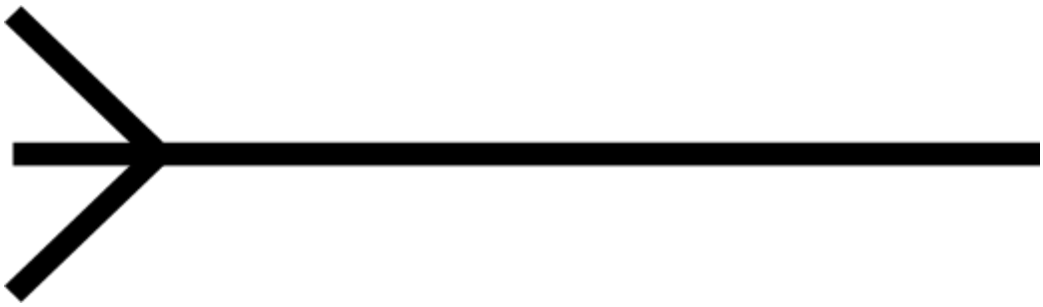
        - Dedicated computer system with a limited or nonexistent user interface and designed to operate completely or largely autonomously from within other machinery
    - **Desktop**
        - Allows a user to carry out a broad range of general-purpose task
    - **Server**
        - Optimised to provide one or more specialised services to network clients

Databases

- **Database** - Structured collection of data
- **Database management system** - A software system that enables the definition, creation and maintenance of a database and which provides controlled access to this database
- **Data model** - a method of describing the data, its structure, the way it is interrelated and the constraints that apply to it for a given system or organisation
- **Conceptual model** - a representation of the data requirements of an organisation constructed in a way that is independent of any software used to construct the database
- **Entity Relationship Modelling**
    - **Entity** - An object, person, event or thing of interest to an organisation and about which data is recorded
    - **Relationship** - An association or link between two entities
    - **Degree of relationship** - The number of entity occurrences of one entity which are associated with just one entity occurrence of the other, and vice versa
    - The are four degrees of relationship
        - **One-to-one**

        - **One-to-many**

        - **Many-to-one**



- **Many-to-many**

- **Normalisation**
  - **Done because**
    - More efficient
    - Lower risk of compromising the integrity of the data
  - **First Normal Form (1NF)**
    - All tables have a primary key
    - All data is atomic (can't be split)
    - Columns with multiple items of data in them must become separate tables
  - **Second Normal Form (2NF)**
    - Must be in first normal form
    - Contains no partial key dependencies, all items not primary or foreign keys must be part of a composite key
  - **Third Normal Form (3NF)**
    - Must be in second normal form
    - Contains no non-key dependencies
- **Relational Databases**
  - **Relational Database**
    - A collection of tables
  - **Attribute**
    - A property or characteristic of an entity (a named column in a table)
  - **Primary Key**
    - An attribute which uniquely identifies a tuple
  - **Composite Key**
    - A combination of attributes that uniquely identify a tuple
  - **Foreign Key**
    - An attribute in one table that is a primary key in another table
  - **Referential Integrity**

        - If a value appears as a foreign key in one table, it must appear as the primary key in another table
- **SQL**
    - **Searching**

```
SELECT <field names>
FROM <tables>
WHERE <conditions>;
```

    - **Ordering Search Results**

```
SELECT <field names>
FROM <tables>
WHERE <conditions>
ORDER BY <field names> ASC/DESC;
```

    - **Inner Join**

```
SELECT <field names>
FROM <table1>
INNER JOIN <table2>
ON table1.columnname = table2.columnname
WHERE <conditions>;
```

    - Select everything using *
    - **Inserting Values**

```
INSERT INTO <table name>
VALUES (<list of values>)
```

    - **Replacing Values**

```
UPDATE <tablename>
SET <newvalues>
WHERE <conditions>
```

    - **Deleting Values**

```
DELETE FROM <tablename>
WHERE <conditions>
```

    - **Creating a Table**

```
CREATE TABLE <tablename>
(
      <fieldname1> <type1>,
      <fieldname2> <type2>,
      ...
      <fieldnameN> <typeN>
)
```

    - CHAR type must have a length, **CHAR(25)**
    - Dates are in quotes
    - **DROP** deletes a database, table or user
    - **ALTER** used to add, delete or modify columns in an existing table

<u>Real Numbers</u>

- **Errors**
    - Precision Error - When precision is lost when a number is stored
    - Absolute Error - Difference between the actual number and the number stored

    - Relative Error - The absolute error divided by the actual number
    - Cancellation Error - Loss of precision when adding or subtracting numbers of widely different sizes
    - Underflow - When the result of a calculation is too small to be stored, a zero is stored instead
    - Overflow - When the result of a calculation is too large to be stored
- **Advantages of Normalised Numbers**
    - Maximises the precision a number can be stored at
    - Unique representation for each number
- **Special Numbers**
    - Smallest Negative

      1.0111111 e=1000
    - Smallest Positive

      0.10000000 e=1000
    - Largest Negative

      1.00000000 e=0111
    - Largest Positive

      0.11111111 e=0111
    - Infinity

      0.00000000 e=11111

<u>Problem Solving</u>

- **Algorithm** - A sequence of unambiguous instructions for solving a problem (can be represented as a Turing machine program).
- An algorithm is said to be **Correct** if for any input returns the correct output
- **Abstraction** Representation that is arrived at by removing unnecessary details
- Two different types of abstraction
    - **Generalisation** - Grouping complex things into simple categories
    - **Representation** - Removing details until it is possible
- **Interface** A boundary between the implementation of a system and the world that uses it
    - Provides abstraction
- **Complexity of a problem** the worst-case complexity of the most efficient algorithm which solves the problem
- **Basic operation** An operation that contributes most to the total run time, this is a single step in the algorithm
- **Comparing Algorithms**
    - Algorithms can be compared by expressing their complexity as a function of their complexity as a function relative to the size of the problem
    - Algorithms can be compared based on their
        - Time complexity
        - Space complexity
    - **Big-O notation** is used to express the complexity of a problem
        - Linear time - $O(n)$
        - Polynomial time - $O(n^k)$, where k is a constant
        - Exponential time - $O(2^n)$

    - **Finite State Machine**
        - **Halting state** - A state with no outgoing transitions
        - **Halting Problem** - Is it possible in general to write a program that can tell, given any program and its inputs and without executing this program, whether the given program with its given inputs will halt?
        - Types of machine
            - **Moore Machine**
                - A finite state machine that determines its output from the present state only
                - The output is in the states
            - **Mealy Machine**
                - A finite state machine that determines its outputs from the present state and from the inputs
                - The output is on the transitions
        - **Finite State Automata**
            - Finite state machines without output
    - **Turing Machines**
        - Numbers are represented in unary, 1 = 1, 2 = 11, 3 = 111, ect.
        - A finite state machine that controls one or more tapes, where at least one is of infinite length
        - A universal Turing machine is one that can simulate any other Turing machine
        - A machine performs one of the following each interaction
            - Alter the tape, writing a symbol or erasing it
            - Move the tape one square left or right
- **Backus-Naur Form**
    - A notation for expressing the rules for constructing valid strings in a regular language
    - <term> - Is a non-terminal symbol, because it is surrounded by angled brackets
    - ::= - This means consists of or is defined as
    - | - This is used to separate alternatives.
- **Reverse Polish Notation**
    - Post-fix notation, the operator comes after the operands
    - Eg. 6 7 + instead of 6 + 7
- **Types of problem**
    - **Non-computable** - An algorithmic problem that admits no algorithm
    - **Decision problem** - A yes/no problem
    - **Decidable** - A decision problem that has a yes/no solution
    - **Undecidable** - A non-computable decision problem
    - **Tractable** - A problem that has a reasonable (polynomial) time solution as the size of the input increases
    - **Intractable** - A problem with no reasonable (polynomial) time solution
    - **Heuristic** - An approach that uses know-how and experience to make informed guesses that assist in finding a polynomial approximation for an intractable problem

Communication and Networking

- Data transmission is the moving of data from one place to another
- **Serial data transmission** is when single bits are sent one after another the a single wire
    - Used in long distance transmissions because only one wire is required and therefore regenerating the signal strength is easier
- **Parallel data transmission** is when bits are sent down several wires simultaneously
    - Used in short distance data transmission, because it is difficult to keep the voltages in the wires the same, this is called **skew**
    - It can be difficult to switch transmission to different paths
- **Baud rate** is the rate at which signals on a wire may change
    - **1 baud** is one signal change per second
- **Bit rate** is the number of bits transmitted per second
- **Bandwidth** is the range of frequencies that a transmission medium can carry
- The higher the bandwidth of the transmission system, the higher the bit rate that can be transmitted over the system
- **Latency** is the time delay that occurs between an initial request and the moment the first effect begins
- **Asynchronous data transmission** is when the sender and the receiver are not synchronised unless exchanging data
- The receiver cannot predict when the transmission begins
    - The sender send a start bit, that allows the receiver to synchronise their clock with the sender's
    - The sender ends the transmission with a stop bit, this is the opposite of the start bit (if the start bit is a 1 then the stop is a 0 and vica versa)
- In a parity system
    - The sender calculates the correct parity bit and ats this to the send data
    - The receiver calculates the parity of the data and check it against the send parity bit
- A **Communications protocol** is a set of rules agreed between two devices to ensure communication is successful
- **Handshaking**
    - The sender and receiver exchange signals to establish that the receiver is connected and ready to receive
    - The sender coordinates the sending of the data, informing the receiver that it is sending
    - The receiver indicates it has received the data and is ready to receive again
- **Baseband system** - A system that uses a single data channel in which the whole bandwidth of the transmission medium is dedicated to one data channel at a time
    - Used in LANs where they offer high performance at a low cost
- **Broadband system** - A multiple channel system in which the bandwidth of the transmission medium carries several data streams at one time
    - Used for long distance data communication, as it is more cost effective to share the medium
- **Local Area Network** - A series of linked computers in close proximity to each other

    - LANs are useful for groups of people working in close proximity, giving them the ability to share printers and transfer files easily
    - Used in businesses and organisations so that file transfer and resource share can be done easily
- **Wide Area Network** - A set of links that connect computers and LANs that are not in close proximity
    - WANs are useful for teams of people spread across a large geographical area that need to share files and resources
    - A multinational corporation or large international organisation that requires its members to access remote resources
- **Internetworking** - when two LANs are interconnected by a WAN
- **Network Topology** - The shape and structure of the connections that connect devices on a network
- **Network Adaptor** - Used to communicate on a network
    - Data received from motherboard into buffer, this data is called the block
    - This block is then given a checksum, source and destination address
    - The is now called a frame and is sent along the transmission medium
- **Bus topology**
    - All computers are connected to a single cable
    - When signals are transmitted voltage pulses propagate the length of the wire
    - Speed of data transmission on a bus network degrades as use gets heavier
    - **CSMA/CD** (Carrier Sense Multiple Access with Collision Detection) has the following rules
        - If the bus is quiet, transmit a frame
        - If the bus is busy, continue to listen until the bus is idle then transmit immediately
        - While transmitting, monitor the bus for a collision. If a collision is detected, transmit a jamming signal to let all computers know that the has been a collision
        - After transmitting jamming signal, wait a random amount of time, then attempt to transmit again, starting from step 1.
- **Switched Ethernet**
    - Uses a star topology, with nodes connected to a central switch
    - When data is transmitted it goes through the switch, which will do one of two things
        - If the bus is free it transmits the data to the other node
        - If data is already being transmitted, the switch will store the data in a buffer and transmit it when the network is free
- **Segmentation**
    - Bus networks are often split into parts called segments as each computer that is added to the network slows it down
    - Segments are connected to one another by bridges and routers
    - **Bridges** contains a table of the hardware addresses of the computers
    - **Routers** contains a table of the IP addresses of the computers
    - Segmentation reduces collisions
- **Star Networks** have a central hub with each computer connected directly to this hud

- **Bus advantages and disadvantages**
    - Can easily be expanded
    - If the backbone breaks the entire system goes down
    - Data must be transmitted to all computers in the network
- **Star advantages and disadvantages**
    - All data must pass through a central server, creating a possible bottleneck
    - Data only has to pass through one node reducing the time required to transmit data
- **Thin client computing** - All processes take place in a central server
    - Clients are dumb terminals, with little or no processing power or local hard drive
- **Peer-to-peer networks** - A network that has no dedicated servers. All computers are equal, so they called peers
- Peer-to-peer networks are used when
    - There are fewer than 10 users
    - The users are all located in the same area
    - Security is not an issue, users may act as their own administrators
    - The network will have limited growth over the foreseeable future
- **Server-based networks** - A network in which resources and other functions are provided by dedicated servers
- **Web 2.0** - software that becomes a service accessed over the internet
- **Web services** - self-contained, modular applications that can be described, published, located and invoked over a network
- **SaaS (software as a service)** - a model of software deployment where an application is hosted as a service provided to customers across the internet
- **Web services architecture** - where all components in the system are services
- **Ajax** - A web technology that allows only part of a web pages that needs updating to be fetched from the web server
- **Wireless Networking** - A type of LAN in which devices (computers, mobile phones, tablets) are connected to each other wirelessly
    - Data is transmitted between the devices with radio waves
    - Often use wifi to transmit the data
    - Devices connect to a WAP (wireless access point) that acts like a bridge between the wired network and the wireless network
- **Wi-Fi** - A standard protocol for wireless data transmission
    - The trademark for the IEEE 802.11 standard
    - Used in home and business networks
- **Bluetooth** - A standard protocol for short distance wireless data transmission
    - Uses radio waves at the 2.4GHz frequency
    - Data is chopped into small packets and transmitted over up to 79 different frequencies, allowing for a gross data transmission rate of 1Mbps, this technology is called frequency-hopping spread spectrum (**FHSS**)
- **Radio Based LAN Protocols** - Any protocol for data transmission that uses radio waves to transmit data instead of wires.
- **Router** - A device that receives packets from one host or router and uses the destination IP address to pass it to another host or router

- Routers will pass packets to other routers depending on the packet's destination IP address.
    - Routers are is a hierarchy, with regional and national routers
- **Routable IP addresses** - An IP address that can be reached by anyone on the internet
- **Non-routable IP addresses** - are used for private networks, they cannot be reached from outside the local network
- **Connecting two LANs by routers**
    - Each LAN has a router, these two routers are connected
    - Each router has two IP addresses, one for each network cards (the router requires one per network)
    - Transmitting data
        - The sender knows the receiver is outside of the network so sets the destination hardware address to be the router, but the IP address in the actual destination
        - The router changes the hardware address to be the recipient's network's router's address. The IP address is unchanged
        - The new router changes the hardware address to the actual recipient's
- **Gateway** - A device used to connect networks using different protocols so that information can be passed from one system to another
    - LANs are connected to the Internet through gateways
- **Subnet mask**
    - Defines the size of the network
    - Sender ANDs subnet mask with their own and the destination IP address, resulting in the sender's and the destination's network ID
    - Networks IDs compared if the same then on the same subnet, if not then the packet must be send to the router.
    - Usually 255.255.255.0
- **Gateway**
    - The IP address of the machine that connects the computer to the outside
    - In a LAN this is the internal IP address of the router that connects the LAN to the WAN
    - Can change the protocol of packets
- **DNS servers**
    - A server that maps domain names to IP addresses.
- **Routing**
    - Sender ANDs subnet mask with their own and the destination IP address, resulting in the sender's and the destination's network ID
    - Networks IDs compared if the same then on the same subnet, if not then the packet must be send to the router.
    - Router identifies the destination is not on the LAN so sends it to the gateway
    - The gateway may first have to change the protocol of the packet, and then passes it onto the internet
    - Each router selects the best path for the packet based on router tables
    - Packet passed from router to router, it's source and destination MAC addresses change each hop

    - Packet works its way up the router hierarchy, until passed to national router then transferred internationally and then back down the hierarchy in the other country
    - Reaches the correct gateway which passes to the subnet's router which passes packet to destination.

Internet Security

- **Virus** - Malicious self-replicating programs which attach themselves to other programs
- **Spam** - Unsolicited junk mail
- **Worm** - Malicious self-replicating programs which replicate across networks using security vulnerabilities
- **Remote Login** - Ability to login into another computer via the Internet
- **Trojan** - A program that hides in or masquerades as a desirable software, but attacks computers it infects
- **Phishing** - Attempts to get users to divulge personal information
- **Pharming** - When a phisher changes DNS server information so that customers are directed to another site
- **Firewall** - A hardware device or program that controls traffic between the internet and a private network.
    - This can do this by
        - **Packet filtering** - The firewall analyses the packets against a set of filters and allows them through or blocks them accordingly
        - **Proxy server** - When a user in the network requests information from the internet, the proxy server retrieves the information then passes it on to the requesting computer
            - The host and user's computers are never in direct contact
    - Can be hardware or software
- Threats can be addresses by
    - Improving code quality
    - Monitoring systems
    - Protecting the system from attacks (eg. firewall)
- **Encryption** - Using an algorithm and a key to convert message data into a form that is not understandable without the key to decrypt the text
- **Plain text** - Message before encryption
- **Cipher text** - Message after encryption
- **Symmetric encryption** - If the cipher text can be decrypted by knowing the encryption algorithm and the encryption key
    - If the language and relative frequency of letters in that language's text are analysed the cipher can be broken
- **Asymmetric encryption (public key encryption)** Both people who want to communicate have two keys a public (known to everyone) and a private (secret)
    - A message encrypted using the private key can be decrypted using the public key
    - A message encrypted using the public key can be decrypted using the private key

    - Best known system known as RSA, works because it is very difficult to factorise very large numbers
- **Digital signatures** - Is a method to ensure that message is from the correct person
    - A hash (or digest) is created from the message
    - Digest encrypted using sender's private key
    - Recipient decrypts the digest using the sender's public key (ensures that the sender produced the digest)
    - Recipient hashes the message and compares it to the digest
        - If they are the same then the message is unaltered
- A digital certificate is provided by a certificating authority and confirms that public key being distributed is genuine
- **Antivirus software** - A scan is performed on files and programs comparing them against a dictionary of known viruses
    - If a virus is found it is deleted
    - The dictionary must be regularly updated to detect new viruses
- **The three As of computer security procedures** are:
    - **Authentication** - to verify that a user of a computer system is a legitimate user. Methods include passwords, biometric data and digital signatures
    - **Authorisation** - The level of permissions a user has and controlling what resources they can access. Used to keep data secret from unauthorised persons
    - **Accounting** - Keeping logs to make it easier to identify parts of the system that have been compromised in the event of a security breach

<u>Server Side Scripting</u>

- **Server-side Script** - A program that is executed by the server when a web page is requested, the output of the program is a webpage that the server returns to the client.
    - Scripts are prefered to executables because their code can be examined for viruses. It is harder to bring down a website using scripts
- **Web server extension** - A program that extends the functionality of the web server and allows it to generate content at the time of the HTTP request
- **Common Gateway Interface** - A gateway between the web server and the web server extension
    - Tells the server how to pass requests to the web server extension
    - Tells the server what to do with information from a web server extension
- **GET method** - Requests data from a specific source
    - Data to be sent to the server is appended to the URL, after a ?
    - Eg. Get /example.asp?name=Cameron
- **POST method** - Data is passed to the server in the body of the message not the URL
    - Eg. Post /example.asp
- **Dynamic web page content** - Content that is generated when the web browser request is received
- **Accessing a database**
    - Create a connection
    - Select a database

    - Perform a database query
    - Use the returned data
    - Close the connection
- **Example scripts**
    - `Variable = Request(“Variable_Name”)`
        - This retrieves the values stored in `Variable_Name` from the web page using POST/GET methods and stores in the variable `Variable`
    - `Response.Write(“Text” + Variable)`
        - Outputs the value held in `Variable` back to the client
    - `Variable = ExecuteSQL("SQL query”)`
        - Query the database (specific)