Model Answers

Algorithmic representation

- Functions vs procedures:
    - **Function:** “Use a function for [task] because it should return the value of … so that it may be used for other modules of the program, such as …”
    - **Procedure:** “Use a procedure for [task] because it does not need to return any value to the calling program”
- Pseudocode:
    - When reading data from a file:
        - “OPENFILE [file name] FOR READ/WRITE/APPEND”
        - READFILE [file name], [variable]
            - Basically readline()
            - Variable (string data type) is assigned the value of the items in the line that was read.
        - The function “EOF (file name)” is used to test whether there are any more lines to be read, and it returns a Boolean type.
        - “WRITEFILE [file name], [data]” writes/appends ‘data’ to the file.
        - “CLOSEFILE [file name]” is used to close the file.

- Iteration:
  - ```
    PROCEDURE INSERTIONSORT(arr: ARRAY)
        FOR index ← 1 TO LENGTH(arr)-1    //base-0 indexing
            unsortedValue ← arr[index]
            current ← index - 1
            WHILE current >= 0 AND unsortedValue < arr[current]
                arr[current + 1] ← arr[current]
                current ← current - 1
            ENDWHILE
            arr[current+1] ← unsortedValue
        ENDFOR
    ENDPROCEDURE
    ```
  - Both ends of a range are inclusive in pseudocode, and the first item has index 1, not 0 (one-based indexing).
  - **Post-condition (executed at least once):** "REPEAT … UNTIL [condition]"
  - **Pre-condition:** "WHILE [condition] … ENDWHILE"
- Object-Oriented Programming
  - Methods and attributes in pseudocode are assumed to be public unless otherwise stated.
  - To show inheritance: "CLASS cat INHERITS pet"
- The "INT()" function always rounds down.
- The "RAND (x)" function returns a random number in the range 0 to x, not inclusive of x.
- DIV is the pseudocode equivalent of "//" (float/integer division) and MOD is the pseudocode equivalent of "%" (remainder).
- "Not equal to" is written as "<>".
- "IF <condition> THEN"
- Comments should be indicated by 2 forward slashes ("//").
- "DATE" is considered a data type in pseudocode.
- You should declare all variables explicitly like this: "DECLARE counter: INTEGER".
- Use "INPUT" and "OUTPUT" to input and output data. You can use commas to output multiple items, like "print()" in Python.
    ```

    - When declaring constants, only literals can be used as its value; you cannot use a variable, another constant or an expression.
        - e.g. “CONSTANT HourlyRate = 6.50”
    - To declare arrays (can only have one type of data inside them, unlike Python lists):
        - **1-dimensional**: “DECLARE nameOfArray: ARRAY [<lower>:<upper>] OF <data type>.
        - **2-dimensional**: “DECLARE nameOfArray: ARRAY [<lower1>:<upper1>, <lower2>:<upper2>] OF <data type>.
    - You can define a procedure like this: “PROCEDURE [identifier] (parameters, if any) … ENDPROCEDURE”. Then, call it like this: “CALL [identifier] (arguments required)”
    - You can define a function like this: FUNCTION [identifier] (parameters, if any) … ENDFUNCTION”. The keyword “RETURN” should be in it to return the value.
        - Similar to Python, once a value is returned, the rest of the lines in the function code are not executed.
    - You can use “<—” (preferred) or “=” to make assignments.
    - To concatenate strings, use “&”, not “+”.
- Flowcharts
    - Use complete names for all variables.
- Trace tables
    - Headings are usually given.
    - **Purpose:** “Used to go through the algorithm line by line, simulating its execution manually and tracking the values of variables and data structures in each step. If the output matches the expected output for all the test cases in the test plan, the algorithm is deemed to be correct.”

Searching algorithms

- Advantages of linear search:
    - Takes less time to attach new items as they can simply be appended at the end of the list
    - Takes less time to delete items as the array does not need to be sorted again after deleting; the item can just be deleted from the location and the location can then be marked as empty
- Disadvantages of linear search:
    - Takes longer to search (especially for long arrays)
        - There is no indication of where to start, and since the data is likely not in order, the entire array must be searched to confirm if the search item is not in the array.
        - **Time complexity:** O(n) — each element is searched one at a time, from the first to the last, so up to n comparisons may be needed to find the element/determine that it is not in the list.
- Advantages of binary search:
    - Takes lesser time to search (i.e. more efficient) due to the continual halving of the array leading to fewer searches
        - **Time complexity:** O(log n) — the list is halved during each comparison and the appropriate half (that may contain the search item) is searched, so up to log(n) comparisons may be needed to find the element or confirm its absence.
- Disadvantages of binary search:
    - Requires the array to be sorted before searching
    - Takes longer to insert new items as the array needs to be sorted after inserting a new item
    - Takes longer to delete items as the array may need to be sorted after deleting an item
- Consider the frequency of the processes you need to do to decide whether to use linear search or binary search.

Sorting algorithms

- To prove that a certain sorting algorithm is being used, use the characteristics of the algorithm.
    - **Insertion sort:** “After each pass, 1 element in the unsorted part is inserted into the correct position in the sorted part.”
- In-place or not:
    - **In-place:** “Extra memory is not used, so it is in-place.”
    - **Not in-place:** “Extra memory is used in the sorting process [give example], so it is not in-place.”
- Explain the worst-case time complexity:
    - “In the worst case scenario, which is …, ‘**[insert line]**’ runs through … times and ‘**[insert line]**’ runs through … times. So, the program runs __ times and the highest power is __, so the worst-case time complexity is __.”
- Explain the best-case time complexity:
    - “In the best case scenario, which is …, ‘**[insert line]**’ runs through … times and ‘**[insert line]**’ runs through … times. So, the program runs __ times and the highest power is __, so the best-case time complexity is __.”
    - e.g. for Quicksort: “The array is halved in each step, leading to logarithmic depth in the recursion tree, and the partitioning step takes linear time.”
- Describe how **[sorting method]** can be used to sort an array:
    - Use a table to describe the method.

Action	Result
e.g. assume the 1st value from the left is sorted	*e.g. 13*
e.g. Compare the next value, 3, with the elements to its left to find its position, then insert it there	*e.g. 3, 13*

- Factors to consider when choosing a sorting method:
    - Use of space — concerns the size of the list, the memory space and even the number of disk drives available (for external sorting)
    - Use of computer time — concerns the number of comparisons and the number of swaps made
    - Programming effort — concerns the effort to write the program
        - *The last factor to consider:* If the time complexity of the algorithm is only marginally better, or the program is only to be run once or twice, then it is not wise to spend a long time investigating many sophisticated algorithms that only save a few seconds of computer time.

Programming elements

- Why is this a [syntax/logic/runtime] error:
    - “Syntax error occurs when program statements cannot be understood because they do not follow the rules of the programming language. In this case, [insert error and explanation].”
    - “Logic error occurs when there is an error in the design of the program. In this case, [insert error and explanation].”
    - “Runtime error occurs when a mistake is made in the processing algorithm, or as a result of external factors not catered for by the program, like lack of memory. In this case, [insert error and explanation].”
- Recursive function definition in context:
    - “A function that calls itself in its definition (line __ of the function), with a stopping condition to stop calling itself (line __ of the function). Each call to itself must tend towards the stopping condition, and in this case, …”

- Why is … a divide and conquer algorithm?
    - It divides the problem into a number of subproblems that are smaller instances of the same problem.
        - [give example]
    - The solutions to the subproblems are then combined to give a solution to the original problem.
        - [give example]
- What is required from the computer system for recursion:
    - "A call stack is required to store the return address, parameters and local variables."
- What is a parameter:
    - "A variable in parentheses of the function used to hold a value that is sent to the function. It acts like an input value to the function."
- Test plan (presented in table format):

Input (examples)	Purpose	Expected output	Actual output
10, 25, 65, 88	Normal data	Accepted	*(record exactly what was output by the program, for example a screenshot of a sample run of the program)*
-1, 101, 200, -55	Abnormal data	Rejected	
0, 100	Extreme data	Accepted	
-1, 101	Boundary data	Rejected	

OOP

- Class diagrams
    - Include getters and setters for all variables unless the question says otherwise.
    - Include a constructor method for all subclasses.
    - Include the data type of all attributes.
    - Take note of '-' and '+' symbols (representing private and public respectively).

- Explain how encapsulation enables data hiding:
    - “Encapsulation is the process of combining methods and attributes into a single object type. It is used to hide the values or state of a structured data object in a class, preventing other classes’ direct access to them.”
    - *Give examples from the context — usually of (public) methods and (private) attributes.*
- Explain encapsulation in context:
    - “Encapsulation is the encasing of private attributes and public methods in a single object type. For example, **[class]** has private attributes **[insert]** and public methods **[insert]**. This prevents the attributes and behaviours of 1 object from affecting the way another functions.”
- Explain polymorphism in context:
    - “Polymorphism is the ability of a method of the same name to behave differently according to its context. For example, **[method]** does … in …, while it does … in ….”
- Explain inheritance in context:
    - “Inheritance is the ability to create new classes by inheriting the data and operations of existing classes. The shared knowledge of the common traits forms the basis of inheritance relationships, which are usually hierarchical from general to more specific classes. For example, **[insert]** is a subclass of **[insert]**.”
- Private attributes:
    - “So that other classes may not see or change the attributes of the objects, following the information hiding principle — where a class cannot directly access the attributes of another when they are declared private.”
- Public methods:
    - “So that other classes may use methods belonging to another class”

- Software reuse:
    - “Due to encapsulation, an abstract data type can be reused to create multiple objects that share common properties and behaviours. For example, multiple … objects can be made from … class.”
    - “Inheritance allows common methods and data to be in the superclass, so that they do not need to be rewritten in the subclasses. For example, some of the methods of **[subclass]** are written in **[superclass]** as they are also methods of **[superclass]**.”
    - “Polymorphism is the reuse of symbols (operators and names) to apply to different object behaviours. For example, **[method]** is reused in different classes.”
- Easy maintenance of software:
    - Because of inheritance, making a change in the superclass automatically makes the same change in all its subclasses without having to manually do it.
- Instantiation of an object:
    - “Instantiation is to create a new object from a class. The constructor sets up a new object so that it has all the attributes and methods defined in the class, but the object has its own data. For instance, **[class]** has been defined with private attributes like **[insert]** and public methods like **[insert]**.”

Linked list

- Describe an algorithm to insert an element at the end of a linked list/in an ordered linked list.
    - Check if the nextfree pointer (head of the free node list is NULL. If it is, display the error message “No free node left” and terminate the program.

    - Else, let the node pointed to by nextfree be A. Update nextfree to point to the next free node in the list, which is pointed to by A. Copy the new element's data into A.
    - If the linked list is empty (head == NULL), set head to the point to A, and let a point to NULL.
    - **To insert at the start:** Else, let the node that the head currently points to be B. Set the head to point to A, and let A point to B.
    - **To insert at the end:** Else, traverse the linked list from the head by following the pointer of each node until the last node is reached. Let the last node be B. Set the pointer of B to point to A, and set the pointer of A to NULL.
    - **For ordered linked list:** Else, from the head, traverse the linked list using a temporary pointer 'current', comparing each node's data with the new data. Use another pointer 'previous' to keep track of the node just before current. Stop when a node is found that [condition for node to be just after the new data], or when the NULL pointer is reached. Let this node be C, and the node just before it be D. Change the pointer of C to point to the new node, and the pointer of the new node to point to D.
        - If the new node is being inserted as the first node, change the head to point to it.
- Removing a node:
    - Starting from the head of the linked list, use a temporary pointer 'current' to traverse through the list. Use another pointer 'previous' to keep track of the node just before current.
    - Stop when the node that current points to contains the data to be removed. Set previous to point to the node pointed to by current.
    - Set the deleted node's pointer to point to the start of the free space list. Update nextfree to point to the deleted node.

- Counting the number of nodes:
    - Start at the head node of the linked list. Initialise a count variable to 0 and a ‘current’ pointer to the start pointer.
    - Traverse the linked list using the current pointer until the last node (where ‘current’ points to NULL). For each node encountered during traversal, increment count by 1.
    - Return count.
- Combining 2 linked lists:
    - If the start of the first linked list is NULL, point its start pointer (start1) to the start pointer of the second list (start2) and terminate the program.
    - Else, traverse through the first linked list using a temporary pointer ‘current’, until the last node is reached (current.pointer == NULL).
    - Change the pointer of current to point to start2 and terminate the program.
- Splitting a linked list into 2 halves:
    - Run an algorithm to count the number of nodes in the linked list (which is above). If it is greater than 1, find the middle node by dividing the total number of nodes by 2 (round up in the case of an odd number). Else, return an error message “Cannot be split” and terminate the program.
    - Traverse the original list until the middle node using a temporary pointer ‘current’, and use another temporary pointer ‘previous’ to keep track of the node just before current.
    - Update the pointer of previous to NULL, and assign a new start pointer to current (head of the new linked list). Return both lists.

Stack and queue

- What is meant by a LIFO data structure?
    - It means that the data that is inserted last in the data structure will be the first to be removed, so it will stay in the data structure for the shortest amount of time.
- Explain overflow and underflow errors:
    - Overflow error occurs when the stack is full so new items cannot be added into it, but attempts to add data items into it are made.
    - Underflow error occurs when the stack is empty and there is no data item to remove from it, but an attempt to remove data from it is made.
- Explain the push and pop operations:
    - The push operation adds a data item onto the stack. Before adding, the stack must be checked for overflow.
    - The pop operation removes a data item from the stack. Before removing, the stack must be checked for underflow.
- Describe an algorithm to remove an item from a stack and place it in a variable ‘var’:
    - Check for underflow. If it occurs, print an error message “Stack is empty” and exit.
    - Else, read the item on the top of the stack and assign it to ‘var’.
    - Decrement the top pointer of the stack by 1.
    - Return ‘var’.
- Explain what a nested/inner function/subroutine is.
    - A function defined inside another function; the outer function calls the inner one.
    - (Used to organise the code or restrict the scope of the inner function.)

- Explain how a stack can be used by operating systems to process nested functions.
    - A call stack can be used.
    - When a subroutine is called, a stack frame, which contains the return address (where the program should continue after the subroutine finishes), the subroutine's parameters and local variables, is created and pushed onto the stack.
    - When the subroutine calls a nested subroutine, another stack frame is pushed onto the stack. This continues for every new function call.
    - When a subroutine finishes execution, its stack frame is popped from the stack and control returns to the return address stored in the previous frame.
    - The use of the stack ensures that each function's context is preserved and that the program can return correctly.
- For a class stack abstract data type:
    - Private attributes:
        - Base
        - Top
        - Maximum size
    - Public methods:
        - Constructor
        - Getters and setters
        - IsEmpty
        - IsFull
        - Pop
        - Push
- Implementing a stack using a linked list instead of an array:
    - **Advantage:** The stack can grow or shrink dynamically without the need for resizing, unlike an array with a fixed size.

    - Disadvantages:
        - Additional memory is required to store pointers.
        - Accessing or allocating memory dynamically may result in slower performance compared to arrays.
- Why is it preferable to implement a queue as a (dynamic) linked list instead of an array?
    - A linked list does not take up unnecessary space in memory as it dynamically shrinks and grows in size.
    - Priority items can join the queue in any position without needing to shift the elements of an array.
    - Because it has a fixed size, an array may be full, in which case new items cannot be added to it.

Binary Search Tree (BST)

- Describe how a BST can be implemented using arrays.
    - Three 1-dimensional arrays can be used: Left and Right, which are arrays of integers and Data is an array of strings.
    - [draw out the arrays with the data values given if possible]
- Explain how these values should be inserted into an empty BST.
    - The first value to be inserted is put into the root node.
    - Thereafter, each value to be inserted is compared to the value in the root node, moving left if it is smaller and moving right if it is larger.
    - This continues at each level until the new item can be inserted as a new leaf node.

RDB

- Explain how the details of all [A] linked to [B] can be found.
    - From … table, search for … that belongs to [B].
    - With …, get all … from … table.

- Data inconsistency:
    - Exists when different and conflicting versions of the same data appear in different places
    - If data is stored more than once, the 2 or more versions of it may be different.
    - [give example from context]
- Data redundancy:
    - A data organisation issue that allows the unnecessary duplication of data within the database.
    - [give example from context]
- What are the problems associated with data redundancy?
    - **Insertion anomaly:** … cannot be inserted until … is known.
    - **Update anomaly:** Any update on … has to be done on all records containing …, such as …. Any missed records that were not updated will cause data inconsistencies.
    - **Deletion anomaly:** If … is completely removed from the records, … may also be lost from the database although it was valuable information that should have been retained.
- What is a primary key or key attribute?
    - Unique identifier for a record in a table
    - [give example from context]
- What is a composite key?
    - At least 2 fields that form a primary key to identify a unique record
    - e.g. A and B can form a composite key if there is no … ID, because it is highly unlikely for the same … to have the same A and B, so the combined value can form a unique identifier for each ….
- What is a foreign key?
    - A field that matches the primary key of another table, value of the field must exist in the primary key of another table

    - [give example from context]
- What is the difference between a primary and foreign key?
    - A primary key does not allow duplicate and null values, but a foreign key does.
- What is the relationship between **A** and **B**?
    - 1 **A** can … many **B**, so the records of 1 **A** can appear more than once in … table.
    - 1 **A** can … in only 1 **B**, so the records of 1 **A** can only appear once in … table.
- Why is this table not in 1NF?
    - Not all columns contain atomic values; some data values can be broken down further.
    - [give example]
- Describe the advantages of using a RDB for storing data on … instead of a customised software.
    - It is easier to add in fields when needed, as less effort is needed to add in tables and fields to a database compared to reprogramming the software.
    - In a customised software, the same information may be held on several different flat files. This causes data redundancy and makes updating data more time consuming. In comparison, in a database system, all the tables are linked, which makes updating more efficient as data updated in one table will automatically be updated in other linked tables. Redundancy is minimised, if not completely eliminated.
    - Within a database system, users have access to information that was previously held in separate files in other departments or sometimes on incompatible systems. A database provides an easy to use query language that enables users to get instant answers to their queries instead of needing a special program

written by a programmer, which takes very long and so is inefficient.

Other databases

- Should an SQL or noSQL database be used in [context]?
    - noSQL
        - Greater flexibility
            - Schema keeps changing.
            - There are fields which only apply to a small percentage of records.
        - There is a hierarchical relationship between items stored in the database.
        - Large number of data items/potential future scalability — it is easier to scale a noSQL database.
    - SQL
        - Schema is largely fixed.
        - Complex queries are to be performed regularly.
        - Strict data integrity and consistency is required.

Data representation

- ASCII vs Unicode
    - **Character set:** ASCII supports only 128 characters, covering basic English, digits, punctuation, control characters and some symbols, whereas Unicode supports characters from all known languages like Chinese and Arabic.
        - Need to represent more languages due to an increasingly globalised world
    - **Memory:** ASCII uses 7 bits per character while Unicode uses up to 4 bytes per character.

    - **Length of encoding scheme:** ASCII has a fixed-length encoding scheme, using a 7-bit binary value; Unicode has variable-length encoding schemes like UTF-8 and UTF-16.
    - **Software localisation:** Unicode supports software localisation — standard applications can be adapted for use in different cultures by simply changing the language and interface; but ASCII does not as it primarily supports English and a few other Western languages.
- Why are hexadecimal numbers used in computing?
    - They are shorter than their equivalent binary numbers, making them more readable and less error-prone for humans.
    - They also make it easier to spot patterns and identify specific bit values.
- How are characters stored in a computer?
    - Stored as a series of 0s and 1s, where each 0 or 1 is called a bit
    - Any group of 0s and 1s can be used to represent a specific character, which can be a letter, number or symbol.
    - 8 bits are used to store one character.
    - The complete set of characters that the computer uses is called its character set.

Data protection, ethics and social issues

- Describe a backup procedure for …
    - Backup a complete set of … data at regular intervals (e.g. daily), and store it in different media elsewhere.
- Describe an archive procedure for …
    - When … data has been updated significantly, select the … data to archive and compress it to reduce storage used by the file(s).
- Explain the consequences of not maintaining backups.
    - Data may be permanently lost.

    - If data is lost, the digital infrastructure may become unusable for some time while the problem is being rectified, causing costly operational downtime.
- Explain the consequences of not performing archives.
    - System performance may degrade over time as more data is stored in it.
    - It may not comply with legal regulations.
- Explain the benefits of using version control.
    - Having an audit trail provides traceability, allowing programmers to revert to previous stable versions of the program.
    - It is easier for multiple programmers to collaborate in teams, without their work overlapping.
- Explain the need for naming conventions.
    - The readability of files, folders, variables, functions etc. improves.
    - File names are less confusing so they are easier to organise.
- What should the DPO do when PDPA obligations are breached?
    - They should act swiftly to contain the breach by…
        - e.g. Contacting the wrong recipient to request that they disregard and permanently delete the sensitive data wrongly sent to them.
- What are the consequences for an organisation when it breaches PDPA obligations?
    - It may face financial penalties imposed by PDPC.
    - Its reputation may be damaged and consumers may lose trust in it.
- Discuss the ethical responsibilities of a self-driving vehicle developer.
    - Communicate openly and truthfully so that users have a clear understanding of the capabilities, limitations and potential risks of the vehicles.

    - Test rigorously before allowing the general public to use the vehicles to ensure that it meets safety standards, so that human safety is prioritised by minimising the risk of accidents.
    - Upgrade the vehicles' software swiftly when an improvement is made.
    - Optimise the routes the vehicles take to reduce congestion, and ensure that they stop at designated spots that are safe for both passengers and road users.
    - Minimise the carbon footprint and other negative environmental impacts of the vehicles.
    - Comply with traffic laws and regulations, such as keeping within speed limits.
    - Do not leak private data collected from clients, and destroy it after a regulated period of time.
- Discuss the impacts of [technology].
    - Social
        - [+] Better communication, connectivity and collaboration across distances
        - [-] Reduced privacy
        - [+] Access to information and entertainment
        - [-] Mis- and dis-information
        - [+] Increased convenience (e.g. e-commerce)
        - [+] Improved safety through better connectivity and location services
        - [+] Better tracking of health indicators
        - [-] Digital divide due to unequal access
        - [-] Cyber threats (especially due to anonymity)
        - [-] Risk of addiction and worsening mental health
    - Economic/workplace
        - [+] Greater productivity and efficiency due to streamlining and automation

        - [-] Job displacement
        - [+] Creation of new industries and jobs
        - [-] Need to upskill to maintain employability
        - [+] Easier to collaborate
        - [+] More flexible work arrangements (e.g. remote work, WFH)
        - [+] Improved data analysis leads to better conclusions being derived

Web applications

- What are the usability principles of an application, and how can they be implemented in…?
    - Learnability
        - Make the placement of buttons and pages intuitive for new users.
        - Include a short introduction to [site] when the user first uses it to familiarise them with common operations like…
    - Efficiency
        - Implement keyboard shortcuts for common operations like…
    - Memorability
        - Use a standard, familiar navigation layout so that users can easily remember it even if they have not used [site] in a while.
    - Satisfaction
        - Create an attractive user interface with an appealing colour scheme and fonts.
    - Error-handling
        - Display clear error messages when [site] encounters a problem and inform the user on what to do to resolve it.

- Discuss the differences between native and web applications.
    - A native application is faster as it is executed locally, compared to a web application which experiences delays due to time taken in transmitting data over networks.
    - A native application can run without a connection to the Internet, while a web application requires an active Internet connection to run.
    - A web application is updated centrally on the server, so updates are seen automatically on the client's side upon requesting data from the server. However, a native application needs to be updated manually.
    - A native application is installed on the user's device, while a web application is not.

Networking

- What is the role of an Internet registrar?
    - It provides domain name registration services to individuals, businesses and organisations.
    - It acts as an intermediary between customers and domain name registries, facilitating the process of acquiring, managing and renewing domain names.
- Roles of layers in communication between devices over the Internet:
    - Application layer
        - Selects the appropriate protocol for data transmission
        - Formats the text data for communication between the application and the network, ensuring compatibility and readability
    - Transport layer
        - Segments data into packets [for TCP] labelled with sequence numbers to ensure ordered delivery

        - Adds port numbers for routing to the correct application on the receiving device
        - [for TCP] Employs TCP for reliable end-to-end connections — utilises acknowledgements to verify successful packet reception, and trigger retransmission if necessary
    - Network layer
        - Assigns source and destination IP addresses to packets, which are used by routers to forward packets to their destination via the most efficient path (not fixed)
    - Data link layer
        - Assigns MAC addresses to devices, enabling hardware identification for packet delivery
        - Handles physical connections between network nodes
- Router vs Gateway
    - A router routes data packets to their correct destination across 2 or more networks that use the same protocol (e.g. TCP/IP).
    - A gateway routes data packets to their correct destination across 2 or more networks with different transmission protocols and/or data formats.
- How do routers transfer data packets?
    - They read the destination IP address of the packet, determine the most efficient and least congested path through the global network at that time, and hand the packet to the next router on that path.
    - This continues until the packet reaches the destination.
- How does a DNS work?
    - The DNS looks up a domain name and converts to an IP address.
    - It checks its local cache to see if it can find the domain name. If it does, it returns the IP address to the user for the user device to go to the web server.

    - If not, it recursively goes up the hierarchy to the next level larger DNS database to resolve the domain name.
    - Once one of the larger databases finds the corresponding IP address, it returns it to the local DNS server (through the higher level DNS servers), which then returns it to the user's device.
    - The local DNS server caches this IP information for future reference.
- What does a data packet header contain?
    - IP addresses of the sender and destination
    - Protocol being used
    - Packet number in sequence
    - Time to live
    - Hop limit (try to use only 1 of the last 2!)
- How does client-server networking work?
    - Many computers (clients) connect to a powerful central server, which stores, manages and delivers shared resources to them.
    - Each client holds some of its own files and software, but it also accesses resources from the server. This is done by sending requests to it, which processes them and provides resources/data to the client.
    - **Use cases:** Web and file servers, cloud computing services
- Advantages of using cloud-based services:
    - Scalable — resources can be increased or decreased as needed, so users only pay for what they actually use
    - Accessible — data and applications can be accessed from any device and location
    - Cost-efficient — reduces the need for expensive hardware and maintenance costs

- Why does the server need to validate user input again, after the client validates it?
    - Client-side validation is usually simple and basic, so it may be circumvented maliciously. Hence, server-side validation is required to ensure data integrity and security.
    - Client-side validation may be limited due to the client's devices.
- LAN vs WAN
    - In a LAN, computers are situated relatively close to each other, while they are geographically dispersed in a WAN.
    - Computers in a LAN are connected using wireless links, wire cables and fibre optic cables, while those in a WAN are connected using telephone links, undersea cables and satellites.
    - In a LAN, data transfer is faster and more reliable than in a WAN, as it does not involve public connections.
    - The setup costs for LANs are lower than those for WANs.
- Intranet VS Internet
    - The Intranet has restricted access (more secure), while the Internet has global access.
    - The Intranet has less traffic congestion, increasing its bandwidth. This makes it faster and more reliable.
- What is the benefit of using an Intranet?
    - It provides a secure and private website that is accessible only to [insiders].
- Static VS dynamic IP addresses:
    - Static addresses are permanent, while dynamic ones are temporary — a different one can be assigned every time the device connects to the network.
    - Static addresses are assigned manually by the network administrator, while dynamic ones are automatically assigned.
    - Dynamic addresses are more cost-effective than static ones.

- IPv4 vs IPv6 addresses
    - IPv4 addresses have 32 bits, while IPv6 addresses have 128 bits.
    - IPv4 addresses consist of 4 decimal numbers separated by dots, while IPv6 addresses consist of 8 groups of 4 hexadecimal digits separated by colons.
- How does subnetting reduce IP address wastage in a network?
    - A large, rigid block of IP addresses is divided into smaller, customised segments that match the actual size of each subnetwork.
    - This is done by borrowing bits from the host portion of the IP address and assigning them to the device portion.
- Thick- VS thin-client computing
    - Compared to the server, less processing and storage happens on thin-client devices than on thick-client devices.
- How does a VPN work?
    - It establishes a secure, encrypted tunnel over the public internet between a user's device and the organisation's private network.
    - It uses a remote server to prevent data from being intercepted and read by unauthorised people.

Network security

- Improving security using:
    - Firewall
        - Blocks unauthorised external traffic by filtering traffic based on predefined security rules regarding the information in packet headers (e.g. IP addresses, ports)
            - Barrier between internal and external networks
    - IDS/IPS
        - Detect (and block, for IPS) suspicious behaviour and payloads containing malware signatures

    - Encryption
        - Converts data into an unreadable format using cryptographic algorithms, which can only be read with a decryption key
        - Make data unreadable even if it is intercepted
    - (Multi-factor) Authentification
        - Allows access only by authorised, verified people
    - Digital signature
        - Ensures the integrity of [data], providing assurance that it has not been altered during transmission
        - Guarantees the reliability of the source/origin
        - Provides strong non-repudiation, so [source] cannot deny sending [data]
            - *[if applicable]* Provides legal protection
- Limitations
    - Firewall
        - Cannot detect insider threats
        - Encrypted traffic reduces the effectiveness of traffic inspection e.g. malware can be encrypted to pass through it
    - IDS/IPS
        - May generate false positives and negatives
        - Processing and analysing traffic reduces network performance
    - Encryption
        - If the key is not stored securely and thus compromised, intercepted data can be decrypted.
    - Authentification
        - Insider threats cannot be protected against.

- Threats
    - Malware (e.g. worm, virus)
        - Such software deliberately causes harm to the data and/or software held on the server
        - Executes malicious code to steal and corrupt data and files, or encrypt systems for ransom
        - **Solution:** Install and regularly update antivirus software to scan and isolate malicious payloads.
            - **Limitation:** New malware may not be recognised.
    - DDoS attacks
        - Flood the server with fake traffic, overloading the system; its bandwidth and CPU is exhausted so it is no longer accessible to legitimate users
        - **Solution:** Configure a firewall to block sudden traffic spikes and drop excessive requests.
            - **Limitation:** Large-scale attacks may still overwhelm the system.
- Why does [data] need to remain confidential?
    - This is sensitive and private information, which may result in [wrong usage] if it is distributed to people with malicious intent.
- Types of detection systems used in IDS and IPS:
    - Signature-based — performs pattern-matching of known attacks
    - Heuristic/anomaly-based — learns what is normal for the network, and flags deviations from this baseline
- IDS vs IPS
    - Similarity — security mechanisms used to detect and respond to suspicious or malicious activities within a network
    - Difference — IDS generates alerts upon detecting malicious activities that provide administrators with information about them, while IPS takes immediate action to mitigate or stop them (e.g. modify network traffic, block certain IP addresses) and

prevent the potential attack from reaching the intended target(s).

- Digital signature process:
  - [data] is hashed using a mathematical function, resulting in a unique message digest that represents its content.
  - [sender]'s private key is used to encrypt the message digest into a digital signature.
  - The digital signature and [data] are sent to [recipient].
  - At [recipient]'s end, the digital signature is decrypted and [data] is hashed using the same hash algorithm as [sender].
  - The 2 resulting message digests are compared; if they are the same, [recipient] can ensure that [data] has not been altered, ensuring data integrity.
  - This way, the authenticity of [recipient] is also ensured.
- Why should multiple defence tools be used simultaneously?
  - Prevents a single point of failure
  - Different tools tackle different threats, and one tool's limitation can be covered by another (complement each other)
  - Provides a layered, more comprehensive approach to network security