H2 Computing 9569 Theory Notes V8

2.4.3 Errors and Debugging		
2.4.4 Program Testing		
2.5: Object Oriented Programming	**Practice/Context**	**Tick here☑**
2.5.1 Object Oriented Programming (OOP)		
2.5.2 Encapsulation		
2.5.3 Inheritance		
2.5.4 Polymorphism		
3.1-3.2: Data Representation	**Practice/Context**	**Tick here☑**
3.1.1 Number Base Conversion		
3.2.1 Character Encoding		
Other relevant applications		
3.3: SQL Database	**Practice/Context**	**Tick here☑**
3.3.1 Relational Database Management System (RDBMS)		
3.3.2 Keys		
3.3.3 Data Redundancy, Data Inconsistency, Data Dependency		
3.3.4 Normalisation		
3.3.5 Entity-Relationship (ER) Diagram		
3.3.8 SQL Query		
3.3: NoSQL Database	**Practice/Context**	**Tick here☑**
3.3.6 SQL vs NoSQL		
3.3.7 Applications of SQL and NoSQL		
3.3.8 NoSQL Query		
3.3: Data Protection	**Practice/Context**	**Tick here☑**
3.3.9 Data Privacy, Data Integrity		
3.3.10 Methods to Protect Data		
3.3.11 Backup, Archive		
3.3.12 Version Control, Naming Convention		
3.3.13 Personal Data Protection Act (PDPA)		

3.4 Ethics	**Practice/Context**	**Tick here☑**
3.4.1 Code of Conduct		
3.4.2 Impact of Computing		
4.1 Computer Networks	**Practice/Context**	**Tick here☑**
4.1.1 Local Area Network (LAN), Wide Area Network (WAN), Intranet, Internet		
4.1.2 Internet Protocol (IP) Addressing, Domain Name System (DNS) Server		
4.1.3 Communication Protocols		
4.1.4 Circuit Switching, Packet Switching		
4.1.5 Organisation & Topology		
4.2 Web Application	**Practice/Context**	**Tick here☑**
4.2.1 Web Application, Native Application		
4.2.2 Usability Principles		
4.3 Network Security	**Practice/Context**	**Tick here☑**
4.3.1 Information Security Threats		
4.3.2 Network Protection Mechanisms		

Link to Syllabus 9569: Singapore-Cambridge A Level (H2) 9569 Computing Syllabus for examination in 2024

1.1 Algorithmic Representation

1.1.1 Pseudocode

Cheatsheet

Comments	//
1D Array of Integers	DECLARE StudentNames : ARRAY[1:30] OF STRING
2D Array of Integers	DECLARE numbers : ARRAY [1 : 5, 1 : 5] OF INTEGERS
x // y	x DIV y
x % y	x MOD y
None	NULL
x = x + 1	x <- x + 1
x == 3	x = 3
x != 3	x <> 3

if n == 3:
    print(“Three”)
elif n == 4:
    print(“Four”)
else:
    print(“Others”)
	IF n = 3 THEN
	  OUTPUT “Three”
	ELSE
	      IF n = 4 THEN
	            OUTPUT “Four”
	      ELSE
	            OUTPUT “Others”
	      ENDIF
	ENDIF

prize = 0
marble = input()
if marble == "blue":
      print("You won $1")
      prize += 1
elif marble == "yellow":
      print("You won $2")
elif marble == "red":
      print("You won $5")
else:
      Beep()
	prize <- 0
	INPUT marble
	CASE OF marble
	      "blue": OUTPUT "You won $1"
	             prize <- prize + 1
	
	      "yellow": OUTPUT "You won $2"
	             prize <- prize + 2
	
	      "red": OUTPUT "You won $5"
	             prize <- prize + 5
	      OTHERWISE: CALL Beep
	ENDCASE

done = False
while not done:
    n = n + 1
    if n > 10:
        done = True
	REPEAT
	  n <- n + 1
	UNTIL n > 10

while n > 0:
    n -=1
	WHILE n > 0
	  n <- n - 1

	ENDWHILE

for Count in range(1, n+1):
    Sum += Count
	FOR Count <- 1 TO n
	  Sum <- Sum + Count
	NEXT

define myProcedure (n):
      for i in range(n):
            print("Hello world")
	PROCEDURE myProcedure(n : INTEGER)
	      FOR i <- 0 TO n-1
	            OUTPUT "Hello world"
	      NEXT i
	END PROCEDURE
	
	CALL Procedure(n)

def myFunction (hList):
      highest = 0
      for height in hList:
            if height > highest:
                  highest = height
            ENDIF
      return highest
	FUNCTION myFunction(hList : array)
	RETURNS INTEGER
	
	      highest <- 0
	      FOR i <- 0 TO LENGTH(array)-1
	            IF height > highest THEN
	                  highest <- height
	            ENDIF
	      NEXT i
	      RETURN highest
	ENDFUNCTION
	
	OUTPUT myFunction(givenArray)

```
CLASS Pet
      PRIVATE Name : STRING
      PUBLIC PROCEDURE NEW(GivenName : STRING)
            Name ← GivenName
      END PROCEDURE
ENDCLASS
```

Inheritance is denoted by the **INHERITS** keyword; superclass/parent class methods will be called using the keyword **SUPER**, for example:

```
CLASS Cat INHERITS Pet
      PRIVATE Breed: INTEGER
      PUBLIC PROCEDURE NEW(GivenName : STRING, GivenBreed : STRING)
            SUPER.NEW(GivenName)
            Breed ← GivenBreed
      END PROCEDURE
ENDCLASS
```

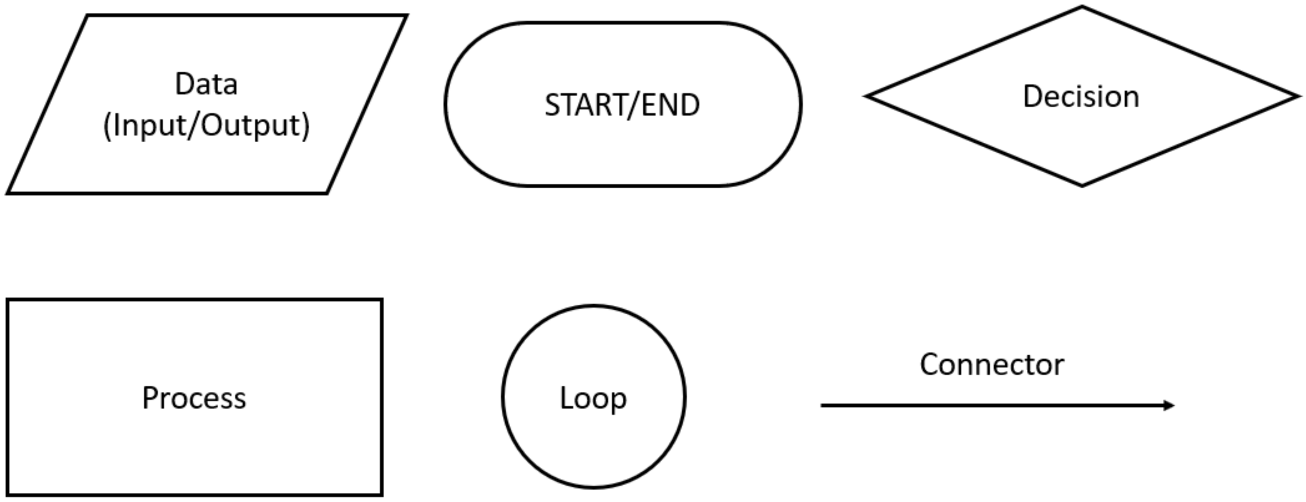
To create an object, the following format is used:
<object name> ← NEW <class name>(<param1>, <param2> ...)

For example:
MyCat ← NEW Cat("Kitty", "Persian")

Link to more pseudocode: pseudocode for teachers

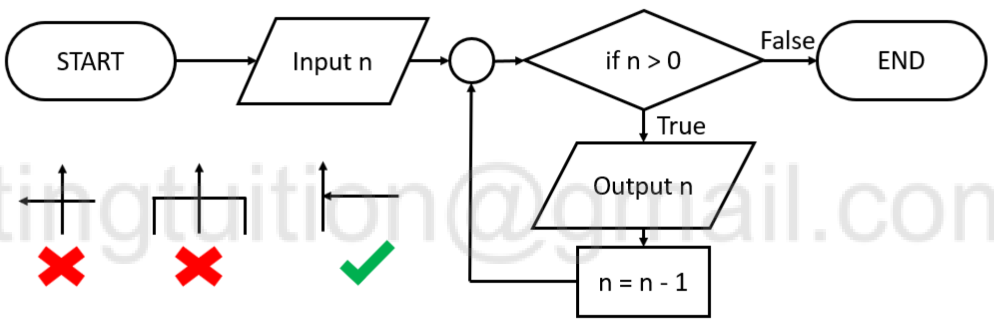
1.1.2 Flow Chart

Flow charts are visual representations of algorithms.



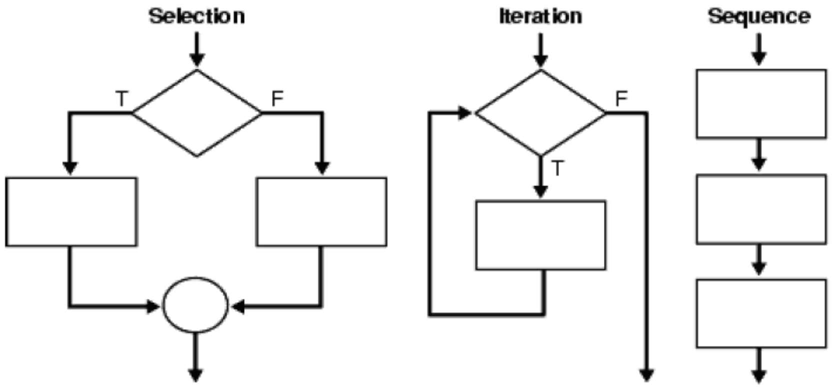
Example

```
PROCEDURE func
  INPUT n
  WHILE n > 0
    OUTPUT n
    n <- n - 1
  ENDWHILE
END PROCEDURE
```



1.1.3 Control Structures

Control Structures	
Sequence	sequential execution of code
Selection	`if, elif, else`
Iteration	`while, for, repeat`



1.1.4 Decision Tables

Decision tables are visual representations of various conditions and their outcomes.

Steps to constructing decision tables

1. Take note of all conditions and actions from the question
2. Draw the table and fill in the conditions and actions given.
3. Fill Y and N in the rules in a binary interval. E.g. 2^0, 2^1 and 2^3 for 3rd, 2nd and 1st rows.
4. Start working on the rules by action by action. For each action, mentally convert the word statement into Boolean statement. Eg 10% discount = Has discount card **AND** Goods totaling more than $100
5. Mark out crosses("X") only for the rules that give **TRUE** for the action. Eg 10% discount = **TRUE**
6. When removing redundancies, compare rules (by column) if any change in one or more conditions still leads to the same outcome. If such a case exists, the condition that changed does not determine the outcome. Thus, they are labelled as dashes ("-") to represent insignificant values.

A shop gives some customers a discount on goods totaling more than $20. The discounts are:

- 5% for goods totaling more than $100
- 5% with a discount card
- 10% with a discount card and goods totaling more than $100

		Rules							
Conditions	Goods totaling more than $20	Y	Y	Y	Y	N	N	N	N
	Goods totaling more than $100	Y	Y	N	N	Y	Y	N	N
	Has discount card	Y	N	Y	N	Y	N	Y	N
Actions	No discount				X	X	X	X	X
	5% discount		X	X					
	10% discount	X							

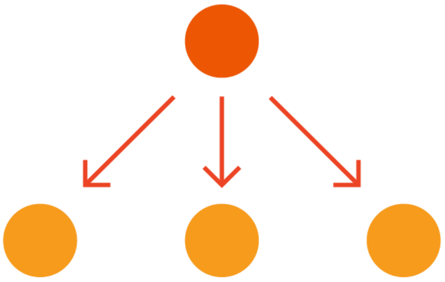
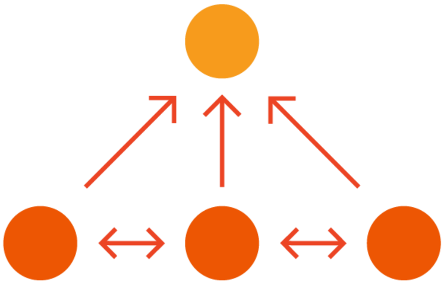
After removing redundancies,

		Rules				
Conditions	Goods totaling more than $20	Y	Y	Y	Y	N
	Goods totaling more than $100	Y	Y	N	N	-
	Has discount card	Y	N	Y	N	-
Actions	No discount				X	X
	5% discount		X	X		
	10% discount	X				

1.1.5 Modular Design

Structure charts are visual representations of the hierarchical structure of modules.

	Top-down approach	Bottom-up approach
Definition	Breaks down complex programs to design its specific functions.	Pieces specific functions together to form complex programs.
Flexibility	Top-down approach is inflexible.	Bottom-up approach is flexible.
Efficiency	Top-down approach is time inefficient.	Bottom-up approach is time efficient.
Objective	Top-down approach ensures that objectives are met.	Bottom-up approach does not ensure that objectives are met.

Example	- Divide and Conquer - Procedural Programming	- Dynamic Programming - Object Oriented Programming (OOP)
Visualisation	**Top Down** 	**Bottom Up** 

Divide and Conquer	Dynamic Programming
Partitions problem into independent sub-problems.	Partitions problem into overlapping sub-problems.
Does not store solutions of sub-problems. When similar sub-problems occur, code has to be re-executed.	Stores solutions to sub-problems. When similar sub-problems occur, previous solutions can be reused.

Modular programming is the process of breaking down/decomposing/subdividing a program into separate sub-programs. It is done through use of **subroutines** and **recursions.**

Subroutines	Procedures	Functions
Allow us to manage and structure our programs by breaking up programs into smaller chunks of code.	Subroutines that do not return values.	Subroutines that allow us to have both input and output. They must return at least one value.

Advantages of subroutine

1. Reduces complicated programs into smaller, more manageable chunks reducing the overall complexity of our program.
2. Appropriate inputs support code reusability and reduce repetitive code.
3. Allows us to make the change in one place and have that change take effect every time the subroutine is called.
4. Easier to test and debug since each subroutine is self-contained

1.2 Fundamental Algorithms

1.2.1 Sorting Algorithms

Bubble sort

Bubble sort performs pairwise comparison of elements and ***performs a swap when they are not in descending (or ascending) order***, such that the smallest (or biggest) element will bubble to the **last array location** after each pass. This continues for n-1 passes (for n elements) or ***until there is no swap within a pass***.

Insertion sort

1. Divide array into two parts: sorted and unsorted. Elements are inserted one by one into their correct position in the sorted section.
2. Consider the first element in the array as a sorted list.
3. Consider the next element in array: if smaller than first element --> swap
4. Consider the third element in the array: swap leftwards until it is in its proper order with the first two elements.
5. Continue until the whole array is sorted.

Merge sort

Merge sort recursively divides an unsorted array into subarrays, until each contains one element. Thereafter, the algorithm repeatedly pairs up the subarrays and merge each pair into a new sorted subarray, until there is only one subarray remaining and that will be the sorted array.

Quick sort

Out-of-place quicksort:

1. A pivot element is selected.
2. The array is partitioned around the pivot, putting every element less than the pivot in a less_than array, elements equal to the pivot in an equal array and elements greater than the pivot in a greater array.
3. Repeat step 1 and 2 for less_than and greater arrays recursively until they can no longer be partitioned into 3 arrays.
4. Eventually, the elements are combined in order by first inserting the elements of less_than, equal and greater

In-place quicksort:

Instead of creating sub-arrays, elements greater than the pivot are swapped with elements less than the pivot. After pivot is added to the middle of the list, it partitions the elements less than itself and the elements greater than itself.

Possible improvement:

- Choose pivot as a randomly generated integer between 0 and size of data-1 instead of choosing the last index. Pivot chosen would be close to median.

Comparing efficiencies of sorting algorithms

Criteria	**Bubble sort**	**Insertion sort**	**Merge sort**	**Quicksort**
Overall comment	Inefficient. but easily implemented due to absence of recursion	Slightly more efficient than bubble sort Easily implemented due to absence of recursion	**Versatile** as it can work well on any type of data sets irrespective of its size and pattern.	**Inconsistent** as efficiency depends on how centred the pivot is in the dataset. Its performance is significantly impacted by worst case scenarios: largest /smallest pivots chosen
Time complexity	- Best: O(n) - Worst: $O(n^2)$	- Best: O(n) - Worst: $O(n^2)$	- Best: O(n log n) - Worst: O(n log n)	- Best: (n log n) - Worst: $O(n^2)$
Stable/ Unstable	**Stable** It doesn't change the ***sequence of occurrence*** of similar elements (i.e., equal elements are ordered in the same order in the original array) and preserves the order in which these elements appeared in the original array.			**Unstable**. It does not preserve the order in which these elements appeared in the original array
Memory usage	Small memory usage		**Large memory usage** as dividing arrays into temporary sub-arrays recursively uses a lot of memory	Out-of-place: Similar to merge sort In-place: Small memory usage as it does not need additional memory space
Sorting almost sorted datasets	Efficient, but slower than insertion sort	**Most efficient**. It can be marginally faster than bubble sort as it only needs to traverse to the correct position instead of going through the entire list elements in a pass.	Inefficient. Even if it is almost sorted, it would still go through the whole process of dividing and merging arrays	**Most inefficient**. It will swap elements within the array based on the pivot value chosen, making the array more unsorted in the intermediate process. This results in higher overhead.
Sorting small datasets	Efficient		Relatively less efficient	Efficient
Sorting large datasets	Inefficient		**Most efficient**	Efficient, but slower than merge sort

1.2.3 Search Algorithms

Linear Search

In a linear search, the program traverses elements in the array in order. The search stops if it finds a match or it reaches the end of the array.

1. Use the number of elements to find the length of the list.
2. Start with the first element.
3. If the current element matches the value searched, return the current index and stop the searching process.
4. If not, iterate to the next element.
5. Repeat Step 3 and 4 until a match is found or the end of the list is reached.

Binary Search

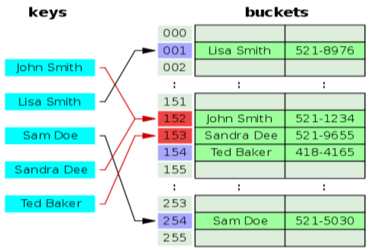
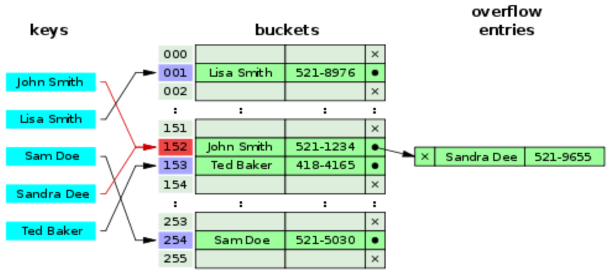
In a binary search of a sorted collection of elements, the collection is ***divided into 2 sections***. The program starts with the element in the middle and ***checks whether the element we are looking for may exist before or after the middle element***. If it exists before, then we continue to search the ***first half*** of the sequence. Otherwise, we should search for it in the ***second half***. The program repeats this ***bisection way of searching*** until a match is found or until the collection cannot be further subdivided.

1. Use the number of elements to find the middle element
2. Check if the searched value is less than, equal to or greater than the middle element.
3. If the value searched matches the middle element, the record is found.
4. If the value searched is less than the middle element, repeat the search with the lower half of the list.
5. If the value searched is greater than the middle element, repeat with the upper half of the list.
6. Repeat Step 2 to 5 until the value targeted is found, or ***list cannot be further subdivided***.

Hash table search

Calculated value is used to mark the position in the table where the data item is to be stored. This enables the data to be accessed directly.

- The calculated address is found by passing a key through a hash function.
- If the location is empty, data is inserted at the location.
- If the location is occupied with different data, a collision has occurred.
- 2 ***collision resolution algorithms*** to handle collision - Open addressing, Closed addressing

	Open addressing (Closed hashing)	**Closed addressing (Open hashing) a.k.a separate chaining**
Description	If collision occurs, the data is not stored in the position it is hashed to. Instead, another location is assigned for it to be inserted. Assigning methods ● Linear probe ● h+1,h+2,h+3 ● Quadratic probe ● h+i2, h+22, h+32  keys buckets John Smith Lisa Smith Sam Doe Sandra Dee Ted Baker 000 001 Lisa Smith 521-8976 002 151 152 John Smith 521-1234 153 Sandra Dee 521-9655 154 Ted Baker 418-4165 155 253 254 Sam Doe 521-5030 255	The data is always stored in the same position it is hashed to. Some other forms of data structure are usually used to store all the data at the calculated position(bucket). Hence, the index in the bucket where the data is stored should be located.  keys buckets overflow entries John Smith Lisa Smith Sam Doe Sandra Dee Ted Baker 000 001 Lisa Smith 521-8976 002 151 152 John Smith 521-1234 153 Ted Baker 418-4165 154 253 254 Sam Doe 521-5030 255 Sandra Dee 521-9655
Advantages	● Hash table has to be as large as the number of keys ● No size overhead - uses only memory allocated for the hash table.	● No theoretical maximum load factor ● Easier delete implementation due to absence of tombstone markers ● Less clustering ● High performance with high load factor due to ability to handle large number of collisions

3 features of good hash algorithm

- Should use all parts of the input data.
- Generate very different hash values for a small change in the input data.
- Hash values it generates should be uniformly distributed across the entire range of possible hash values (to minimise collisions).

Comparing efficiencies of searching algorithms

Criteria	**Linear Search**	**Binary Search**	**Hash table search**
Overall comment on efficiency	Inefficient, as linear search needs to iterate through all items in the list.	Efficient, as binary search reduces the number of items to search through by half after each iteration. Such a divide **and conquer approach** would incur less computational cost and would be faster.	Very efficient as the address of data can be calculated at a constant time.
Time complexity	- Best: O(1) - Worst: O(n)	- Best: O(1) - Worst: O(log n) **(sorting of array increases the overall time complexity)**	- Best: O(1) - Worst: O(n) (if there are too many collisions)
Requirements for datasets	Can be done on all types of data	All elements must be arranged in sorted order (increasing/decreasing)	Can be done on all types of data as long as they are stored in the hash table.
Searching in small datasets	Efficient	Less efficient, considering additional time complexity of sorting	Efficient but requires much effort to implement
Searching in large datasets	Inefficient	Efficient	Most efficient

1.2.5 Time Complexity

However, the time efficiency of an algorithm only describes how the execution ***time scales with the size of data*** but is not an indication of speed for particular data size.

$$logn < \sqrt{n} < n < log\, n < n^2 < n^3 < n^4 < n^6 < 2^n < n!$$

Time Complexity	Examples

$logn$	`def function(n):`
	`    i = 1`
	`    while i < n:`
	`        i *= 2`

$\sqrt{n}$	`def function(n):`
	`    i, j = 0, 0`
	`    while j < n:`
	`        j = i * i`

n^2	`def function(n):`
	`    for _ in n:`
	`        for _ in n:`
	`            n += n`

2^n	`def function(n):`
	`    if n <= 1:`
	`        return n`
	`    else:`
	`        return function(n-1) + function(n-2)`

1.3 Data Structures

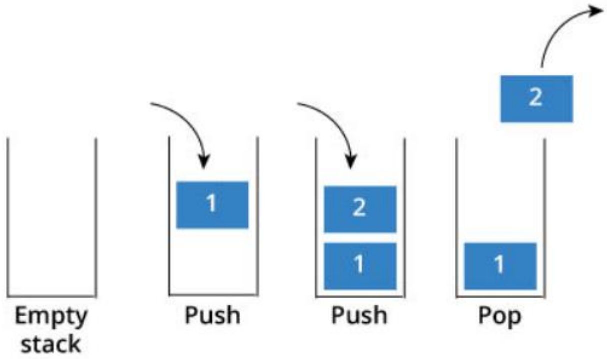
1.3.1 Memory Allocation

	Static Memory Allocation	Dynamic Memory Allocation
Definition	Memory that is allocated cannot be changed.	Memory that is allocated can be changed.
Memory size (flexibility of memory space)	**Inflexible**. Static data structures reserve memory for a ***set amount of data***. This is established by the programmer in advance, even though less memory may actually be needed during program execution.	**Flexible**. The programmer does not need to know about the required size of memory space. Instead, the programmer can **shrink or expand the allocated memory space as required**.
Reuse of memory	Since memory is allocated in **compile time**, memory cannot be reused.	Since memory is allocated in **runtime**, memory can be reused.
Handling small data	Memory is wasted if, for example, the array is declared with space for 100 elements but only 10 elements are added when the program is run.	It only **uses as much memory needed to store the data**. This results in better **memory utilisation**, as the right amount of memory is allocated.
Handling large data	On the other hand, if insufficient space has been allocated, the program would fail to store all the data required.	
Inserting/Deleting from ordered data	Memory locations of rest of the elements must be changed to add a new element between elements	An element can be easily added between elements by simply **re-assigning the pointers.**
Memory allocation	Memory only needs to be allocated once during the creation of the static data structure and only needs to be deallocated once during the deletion of the static data structure.	Each ***insertion*** or ***deletion*** will involve memory allocation or deallocation, which has ***higher overhead*** compared to a static data structure. This might lead to a slower performance.
Search	If the static array is sorted, more efficient binary search can be used instead of linear search	Only linear search is allowed
Accessing elements	Elements can be easily accessed using an ***index position at a constant time***	To access the element, its node must be searched first through ***traversal of the data structure***. Once found, elements can be accessed.
Code complexity	Memory usage of static data structures does not need to be monitored.	Memory use of dynamic data structures has to be monitored so that it does not consume all device memory.
Memory overflow	Cannot use more memory than has been allocated	Can use up all available memory on the computer system, causing programs requiring more memory to hang.

Example	- **Array:** Elements are stored in a ***single continuous/contiguous block of memory***.	- **Linked list:** Nodes are stored in different parts of the memory and are ***linked up using pointers***.

1.3.3 Stack

Stack is an abstract data type (ADT) which inserts and removes items according to the **Last-In-First-Out** (**LIFO**) principle. A user may insert items into a stack at any time, but may only access or remove the last item inserted.



Stack implementation

Stack can be implemented using either **array** or **linked list**.

The basic operations of a stack is to

- push(): add an item to the top of the stack
- pop(): remove and return an item from the top of the stack

1.3.3 Queue

Queue is another ADT which is a collection of items that are inserted and removed according to the **First-In-First-Out (FIFO)** principle. That is, items can be inserted at any time, but only the item that has been in the queue the longest can be next removed. Queue **preserves the order of data**.

DEQUEUE
(First to remove from the front)

FRONT

REAR

ENQUEUE
(Enters the queue from the rear)

Linear queue implementation

Queue can be implemented using either array or linked list.

The basic operations of a queue is to

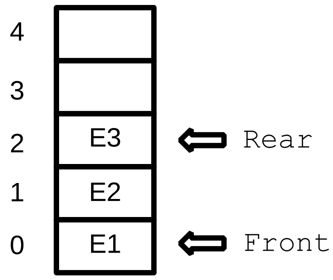
- enqueue(): add an item to the rear of the queue
- dequeue(): remove an item from the front of the queue

Limitation of linear queue:

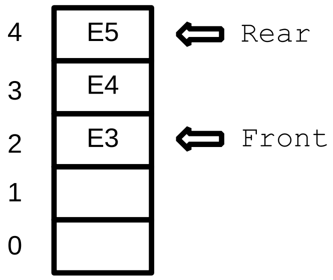
When dequeuing, pop(0) is used to remove the first element from the list. This is inefficient as a loop is executed to shift all elements beyond 0 index to the left, so as to fill the hole in the sequence created by pop(0). Therefore, it always causes the worst-case behaviour of O(n) time.

Improvement:

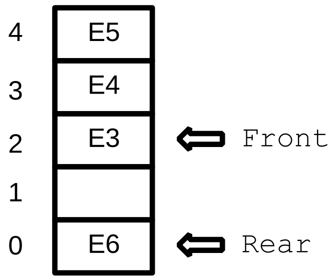
Use ***modulo arithmetic*** to implement a wraparound such that after the *Rear* reaches the last index , it will ***wrap around and point to index 0***. The next insert can then be to the index 0.
A change must also be made correspondingly to *Front* when an element is deleted so that this ***frees up the space also in a circular manner***. This is known as **circular queue implementation**. Circular implementation makes the queue more time efficient and space efficient.



i) inserted 3 elements in the order of E1, E2 and E3 in a Queue

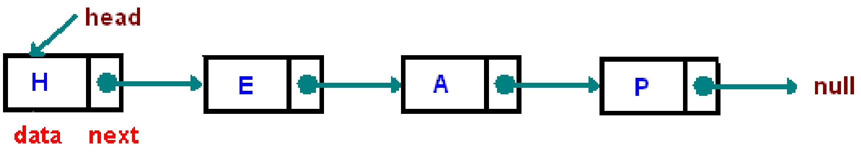


ii) the Queue with some elements inserted (E4 and E5) and removed (E1 an E2)



iii) after inserting E6, the rear arrow is wrapped around and is now below the front arrow

1.3.5 Linked List



A **singly linked list** is a collection of nodes, each containing one pointer pointing to the next node. The start pointer points to the first node, and the pointer in the last node points to null. When the list is empty, the start pointer points to null. With only one pointer, it can only traverse forward along the list.

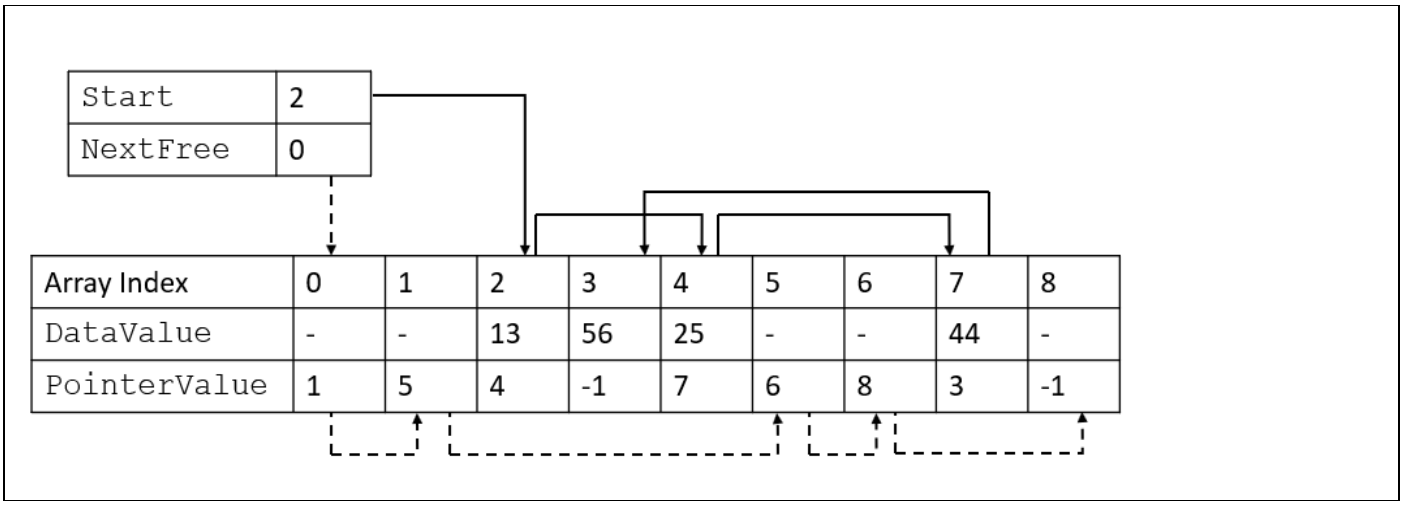
Limitation of a linked list:

Nodes that are deleted will remain in the array and no longer be accessible as no pointer will point to it. This space, which is free, will then be wasted. One way of managing the free space is to keep two linked lists: one for the actual data, and one for the free space.

Improvement:

A **free space list** can be created by linking unused nodes (empty nodes initially). In this manner, new nodes can be created using the unused nodes and nodes that are deleted will be returned to the free space list and can be reused again.

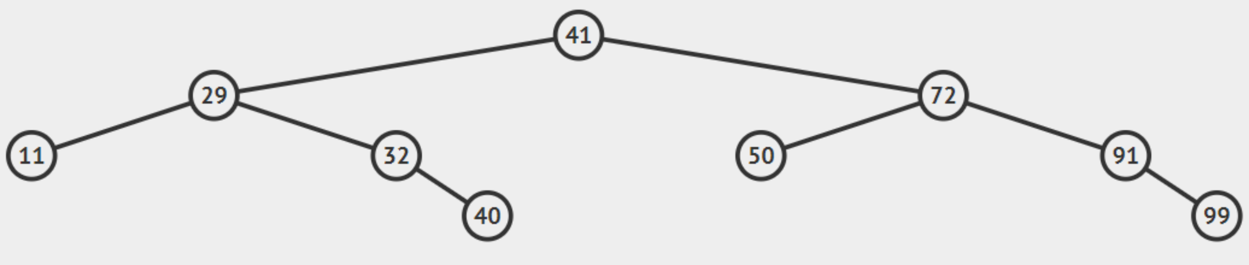
Implementation using free space list



Start	2
NextFree	0

Array Index	0	1	2	3	4	5	6	7	8
DataValue	-	-	13	56	25	-	-	44	-
PointerValue	1	5	4	-1	7	6	8	3	-1

1.3.6 Binary Search Trees



"Tree": An abstract data type which allows each node to have **more than one node** unlike linear data structure	"Binary": The number of child nodes **restricted to 0, 1, or 2**. In its implementation, typically by OOP, a node object will have the attributes: ● data ● left pointer (points to the left child) ● right pointer (points to the right child)	"Search": It automatically sorts the data as nodes are added in the following manner: ● left child: lower data compared to that of its parent ● right child: higher data compared to that of its parent

Searching for nodes in BST

Start with the root node. If the data searched is the data in the root node, it is found. If the data is larger than that of the root node, continue the search with the right child of the root node. If the data is smaller than that of the root node, continue the search with the left child of the root node. The process repeats until the searched data is found or if the leaf nodes are searched and the data is not found.

Deleting nodes in BST

CASES

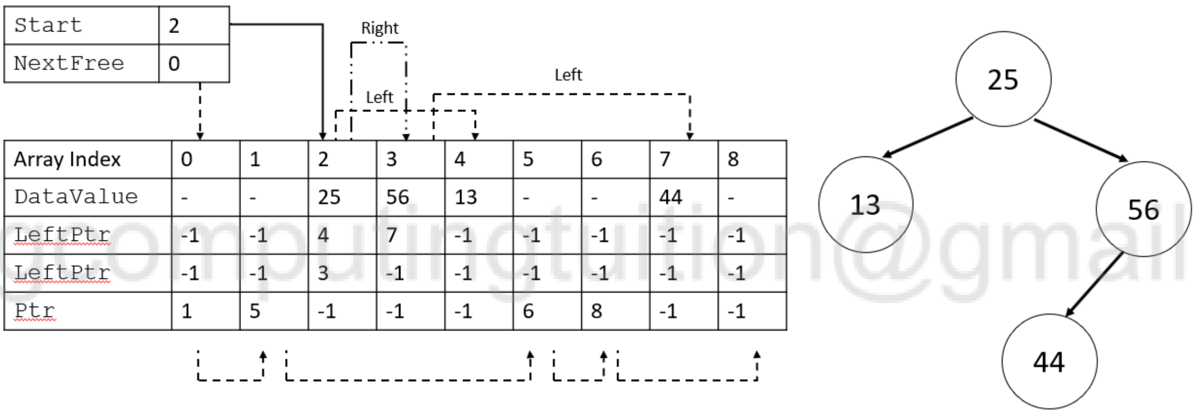
1. No child nodes: Just remove node
2. 1 child node: replace node with child node and its descendants.
3. 2 child nodes: look for the node with the next larger value.

- First, branch right.
- Branch left until there is no left child node.
- Replace the original node with the final child node.

Time Complexity of BST for Search Operation

The height of a BST is the height of its root node. The height also corresponds to the maximum number of comparisons needed when we implement the search method. Similar to the other algorithms that utilise the “divide and conquer” strategy, in the best-case scenario (a balanced tree), the time complexity is O(log n). However, in the worst-case scenario, the time complexity is O(n).

Implementation using free space list



Start	2
NextFree	0

Array Index	0	1	2	3	4	5	6	7	8
DataValue	-	-	25	56	13	-	-	44	-
LeftPtr	-1	-1	4	7	-1	-1	-1	-1	-1
LeftPtr	-1	-1	3	-1	-1	-1	-1	-1	-1
Ptr	1	5	-1	-1	-1	6	8	-1	-1

1.3.7 Traversal

The process of visiting every node in the tree exactly once. In BST, the visitation is done before exploring the left and right subtrees.

- Pre-order Traversal: Node-Left-Right (NLR)
- In-order Traversal: Left-Node-Right (LNR)
  - To display data in an incremental order
- Post-order Traversal: Left-Right-Node (LRN)

Comparing hash table and BST

	Hash Table	Binary Search Tree
Sorting	No operation that allows us to arrange all keys in ascending order.	In-order traversal allows us to display items in an incremental order.
Searching for closest value	Hash tables do not allow us to search for the key that has the closest value to the target.	BST allows us to search for the key that has the closest value to the target.
Memory efficiency	Hash table requires a lot of memory than what is required to prevent collisions	BST requires just the exact amount of memory for nodes that store the data.
Best time complexity of searching	Hash table has the best time complexity of O(1).	BST has the best time complexity of O(logn).
Worst time complexity of searching	Collisions might occur in hash tables, increasing its search time. In that case, the worst time complexity is O(n). **Measure**: Choose appropriate size of hash table based on the size of data to minimise number of collisions	If the BST is unbalanced, the number of iterations of search becomes closer to n. Hence, the worst time complexity is O(n). **Measure**: Implement a self-balancing BST to make sure that it is balanced every time there is a change in the table.

Overview of ADT

Criteria	Array	Python list	Linked list	Binary Search Tree
Type of data structure	Static data structure	Dynamic data structure		
Number of items (size)	Fixed	Variable		
Data type of element	Should have same data type	Can have different data types	Element is known as a node and it contains data and a pointer	Element is known as a node and it contains data, left pointer and right pointer
Memory location	Addresses are contiguous - next to each other/ adjacent in memory		Elements are stored at whatever location that is available and are linked into an ordered sequence using pointers.	
Searching operation	Efficient. Elements can be accessed by an index with a **constant access time**. Hence, an array can use binary search which has a low time complexity of **O(log n).**		Inefficient. Since direct access to elements is not allowed, linked list requires **iteration**. Hence, the search time for linked list is **linear** and has a time complexity of **O(n)**.	The left subtree contains data items that are smaller than the root while the right subtree contains data that is greater. Thus, each time a root is compared with the search item, **the search space is reduced by half**, resulting in time complexity of **O (log n)** which is faster than the linear time of a linked list.
Inserting /deleting operation	Elements are stored at **contiguous memory locations** when an array is used. To maintain the elements in an increasing order, a deletion or insertion operation will result in **extensive shifting of elements within the array**. This increases the time complexity of insertion/ deletion operation		We just need to **update the address stored in the pointers of the nodes affected** during an insertion or deletion operation. There is **no need to change the memory location of any elements.** Hence, it has a low time complexity of insertion/deletion operation when storing elements in order.	

2.2 Programming Elements

2.2.5 Recursion

Recursion function

- Function that is defined **in terms of itself**.
- Calls itself with one or more **similar** but **smaller** subproblems.
- Repeats itself several times and **with each recursive call, the problem is brought closer to the base case**.
- Once the terminating case/ base case is reached, it stops calling more functions. A base case/ terminating case is the smallest problem that can be solved trivially.

Example

```
def summation(lst, count = 0):
    if len(lst) == 0: # base case
        return count
    else: # general procedure
        return summation(lst[1:], count + lst[0])
```

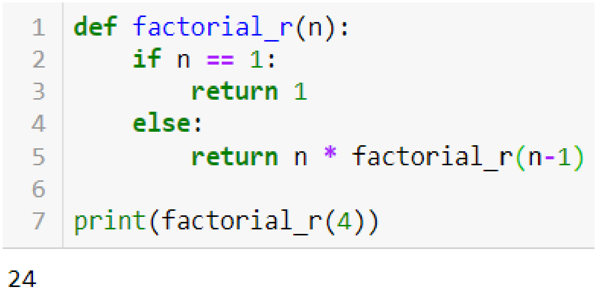
Advantages of recursion

Advantages	Disadvantages
1. More elegant and uses less program code than iterative solutions . 2. Complex tasks can be broken down into simpler sub-problems. 3. Easier to implement when designing a solution to a mathematical problem that is recursive by nature.	1. Repeated recursive calls can incur large amounts of **memory usage** and **processor time** from the multiple function calls and ***storing of return addresses and copies of local or temporary variables***. 2. If the number of calls exceeds a certain limit, the call stack would run out of memory space. This will result in a ***maximum recursion depth exceeded error*** (stack overflow) which causes the program to crash.

2.2.6 Code Tracing

by using 3 diagrams: tracetable, trace tree, call stack

Code:



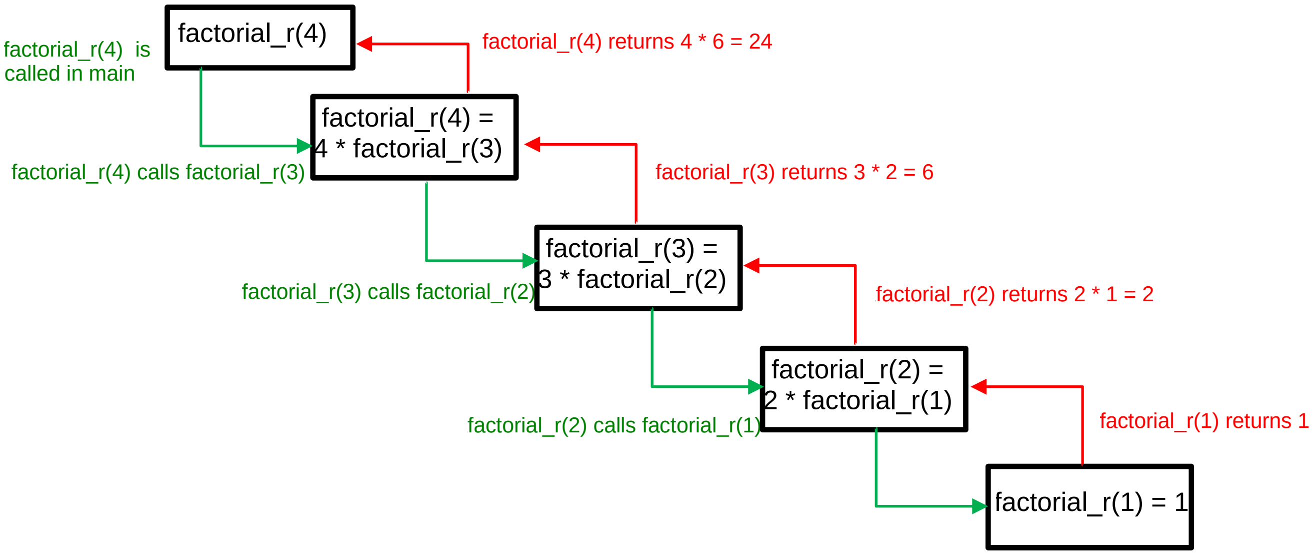
```
def factorial_r(n):
    if n == 1:
        return 1
    else:
        return n * factorial_r(n-1)

print(factorial_r(4))
```

```
24
```

Trace table:

Call number	Procedure call	n	n == 1	Return value
1	factorial_r(4)	4	False	
2	factorial_r(3)	3	False	
3	factorial_r(2)	2	False	
4	factorial_r(1)	1	True	1
3	factorial_r(2)	2	False	2
2	factorial_r(3)	3	False	6
1	factorial_r(4)	4	False	24

Trace tree:


factorial_r(4)
factorial_r(4) is called in main
factorial_r(4) returns 4 * 6 = 24
factorial_r(4) =
4 * factorial_r(3)
factorial_r(4) calls factorial_r(3)
factorial_r(3) returns 3 * 2 = 6
factorial_r(3) =
3 * factorial_r(2)
factorial_r(3) calls factorial_r(2)
factorial_r(2) returns 2 * 1 = 2
factorial_r(2) =
2 * factorial_r(1)
factorial_r(2) calls factorial_r(1)
factorial_r(1) returns 1
factorial_r(1) = 1

2.2.7 Call Stack

- Call stack is used to **keep track of recursive calls**.
- When a recursive call is made, the **return address** at which the function is called and **current contents of the local variables** are stored on the stack as a **stack frame**.
- Each recursive call will push another stack frame to the stack **until the base case is reached**.
- Once the base case is reached, the main will **return to the caller of the function** by **popping off the stack frame**, restoring the return address and contents of the local variables of the caller.
- The process of popping off the stack frame will continue until the **control is back to the first caller to the function**.

1
Return address to factorial_r(2)
2
Return address to factorial_r(3)
3
Return address to factorial_r(4)
4
Return address to main

2.4 Program Testing

2.4.1 Data Validation and Data Verification

Data validation

Data validation is a check that the data entered ***conforms*** with the ***specific set of data requirements of the system*** and ***rejects data considered unreasonable for the program***.

Validation techniques

Range check	e.g. The expiry date of a card must have a month number between 1 and 12, and the date must be later than today's date.
Format check	e.g. A date has to be dd/mm/yyyy
Length check	e.g. A telephone number must have a length of 8 numbers.
Presence check	e.g. ensure that an entry field is not left blank.
Check digit	e.g. Use digits of the identification number to do a calculation to add an extra digit to the end.

UK Syllabus: Data type check, uniqueness check

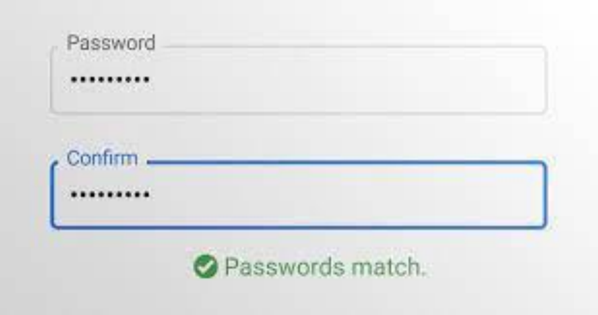
Workflow for validation

- if user provides invalid data, prompt error message and allow re-entry
- until user provides valid data (looping)

Data verification

Data verification is the process of getting the user to ***confirm that the data entered was what was intended*** to be entered.

Verification Methods

Double entry	Entering data twice with the second entry being compared with the first is to ensure that it is accurate.  Password ••••••••• Confirm ••••••••• ✓ Passwords match.
Proofreading data (Visual check)	Someone checking the data entered against the original document.

Modulus 11 Check Digit

<table>
<tr><td><b>Step 1</b></td><td>Multiply each digit of the original number by its weight at that position, and find the sum of these products.
<b>Example:</b> <i>02757</i>
<table>
<tr><td>Position</td><td><b>0</b></td><td><b>1</b></td><td><b>2</b></td><td><b>3</b></td><td><b>4</b></td></tr>
<tr><td>Weights</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td></tr>
</table>
$(0 \times 6) + (2 \times 5) + (7 \times 4) + (5 \times 3) + (7 \times 2) = 67$</td></tr>
<tr><td><b>Step 2</b></td><td>Take the result from step 1 and modulo 11.
<b>Example:</b> $67\ (mod\ 11) = 1$</td></tr>
<tr><td><b>Step 3</b></td><td>Subtract the result from step 2 from 11 .
<b>Example:</b> $11 - 1 = 10$</td></tr>
<tr><td><b>Step 4</b></td><td>If result in step 3 is 10, it is replaced by X. Otherwise, keep the result as it is. This result would be the original number’s check digit, and is added to the far right of the original number.
<b>Example:</b> $02757X$</td></tr>
</table>

Types of errors check digits usually detect:

e.g.

- transposition error (two digits are swapped)
- transcription error

Reasons why the data type of a field storing the code number with its check digit should be a string rather than an integer:

- An integer field will not be able to store a code number correctly if the code number starts with zero because zeros will be omitted and causes error in calculating the check digit.

- The code number may also contain alphabets, making integer an invalid data type for storing alphanumeric characters.

2.4.3 Errors and Debugging

	Syntax error	Run-time error	Logic error
What is it?	Error that causes the translation from source code to machine code to fail.	Error that causes the program **to exit unexpectedly during execution**. It is not detected when the program is compiled but is only revealed when a particular line is executed.	Error that causes a running program to **return an undesired result** but ***does not crash or hang***. The error is caused by a mistake in the program's logic.
When does it occur?	When the translator translates the source code to machine code	When the program is run	When the program is run
Why does it occur?	● Spelling mistakes ● Leaving out a keyword ● Putting a keyword in the wrong place ● Leaving out a symbol, such as a colon, comma or brackets	● Input data has not been validated. ● Running out of memory ● Accessing non-existent objects ● Performing an operation on incompatible types	● Incorrect/incomplete algorithm ● using the wrong variable name ● indenting a block to the wrong level ● using integer division instead of floating-point division

2.4.4 Program testing

Types of test cases

Normal data	Abnormal data	Extreme data
Data follow what is expected during normal use of the program	Data falls outside the set of acceptable values and should be rejected by the program.	Data is at the limit of what the program is designed to cope with or where special handling is required.

2.5 Object Oriented Programming (OOP)

2.5.1 Object Oriented Programming (OOP)

Object-oriented programming (OOP) is a form of programming that groups the data items and the subroutines that operate on the data items in one place called an **object**.

Terms	Meanings
Attributes/Properties	Variables of an object
Methods	Subroutines which access and assign values to attributes
Class	A template for creating objects in a program. It contains attributes and methods
Object	An instance of a class

2.5.2 Encapsulation

Encapsulation is the combining of **private properties (-)** and **public methods (+)** into a class and ensuring private properties are **only accessed/altered by calls to public methods**

Public methods

- **Getter methods**: Subroutines to access the attributes
- **Setter methods**: Subroutines to update values of attributes after instantiation

Properties of encapsulation

- Encapsulation achieves **information hiding** by hiding the internal representation of an object and providing simple public methods for programmers to interact with it. This provides data security and ***prevents unauthorised access or modification*** of the object's internal state.
- Encapsulation achieves **implementation independence** by allowing internal representation of an object to be independent of external code which interacts with it through public methods. Thus, it can be upgraded without changing the codes which use it, promoting **flexibility** and **code reusability**.

Person
-name -mobile_no

+constructor()
+get_name()
+get_mobile_no()
+set_mobile_no(mobile_num)

2.5.3 Inheritance

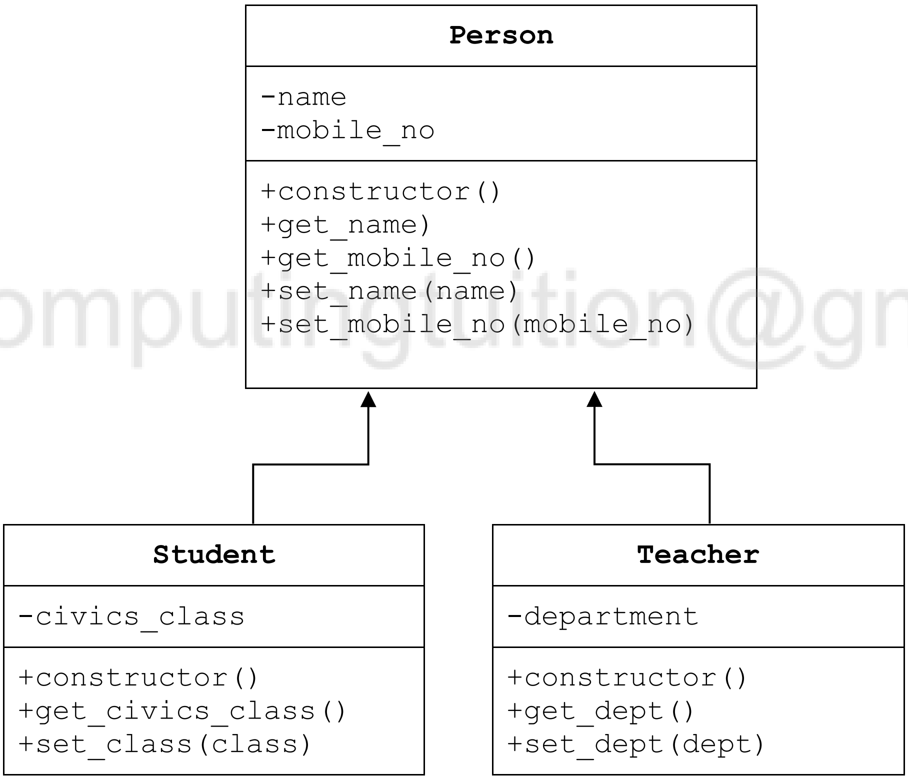
Inheritance is an OOP technique where child ***classes are derived from a parent class*** and inherit all the attributes and methods from the parent.

Properties of inheritance

- Inheritance promotes **software reuse** as developers derive new child classes from the parent class that is already built with attributes and methods they want, such that those modules would be reused instead of having to re-write the whole class from scratch.

Class diagram

* Use **PascalCase** for class names & **snake_case** for attribute and method names
* "-" means private and "+" means public



2.5.4 Polymorphism

Polymorphism is an ability for a method in a subclass to behave differently from that of its superclass despite having the same identifier for the method.

Properties of polymorphism

- Polymorphism utilises **method overriding**, allowing a subclass to provide a specific implementation of a common method in its superclasses.
- Polymorphism allows **code generalisation** as developers can write code that ***operates on a common interface*** and ***handle objects of different types*** without explicitly knowing their specific implementations.

Advantages to OOP

- **Improve code readability**: Program structure is short and concise. OOP allows breaking the program into bite-sized problems that can be solved easily (one class at a time).
- **Improve code reliability**: Avoid code duplication and easier to debug.
- **Reuse quality code**: Reuse existing code which is already tested.
- OOP systems can be easily upgraded from small to large systems.

3.1-3.2 Data Representation

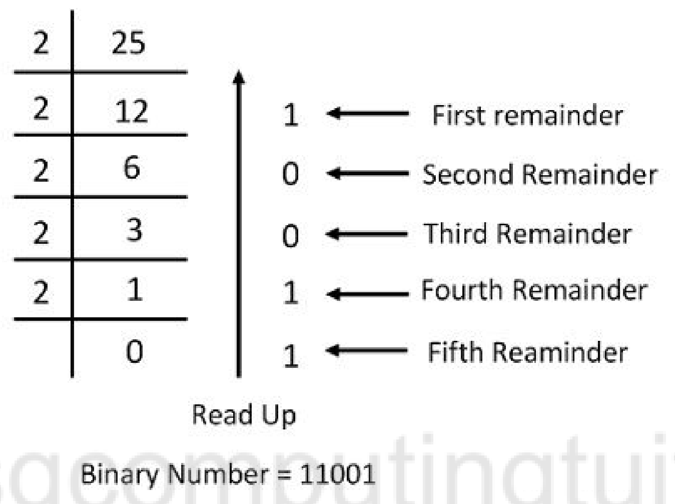
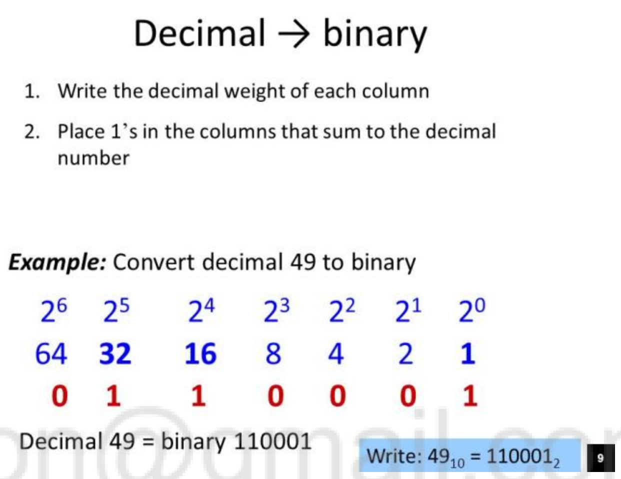
3.1.1 Number Base Conversion

Conversion from base N to base 10

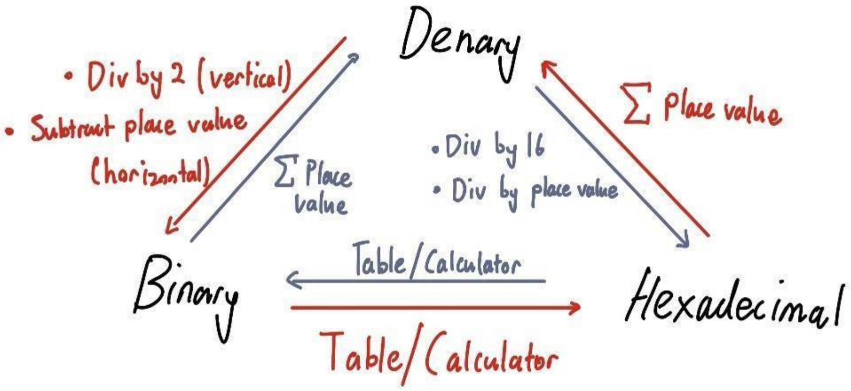
Binary	*1*	*1*	*1*	*0*	*1*	*0*	1
Denary	1×2^5	1×2^4	1×2^3	0×2^2	1×2^1	0×2^0	1×2^{-1}

$$111010.1_2 = 32 + 16 + 8 + 0 + 2 + 0 + 0.5 = 58.5_{10}$$

Conversion from base 10 to base N

Repeated Division	Sum of Weights
 2 25 2 12 2 6 2 3 2 1 0 1 First remainder 0 Second Remainder 0 Third Remainder 1 Fourth Remainder 1 Fifth Reaminder Read Up Binary Number = 11001	Decimal → binary 1. Write the decimal weight of each column 2. Place 1's in the columns that sum to the decimal number ***Example:*** Convert decimal 49 to binary 2^6 2^5 2^4 2^3 2^2 2^1 2^0 64 32 16 8 4 2 1 0 1 1 0 0 0 1 Decimal 49 = binary 110001 Write: $49_{10} = 110001_2$

Place value	256	128	64	32	16	8	4	2	1	Denary
	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	~~135~~ ~~7~~ ~~3~~ ~~1~~ 0
Binary Digit		1					1	1	1	

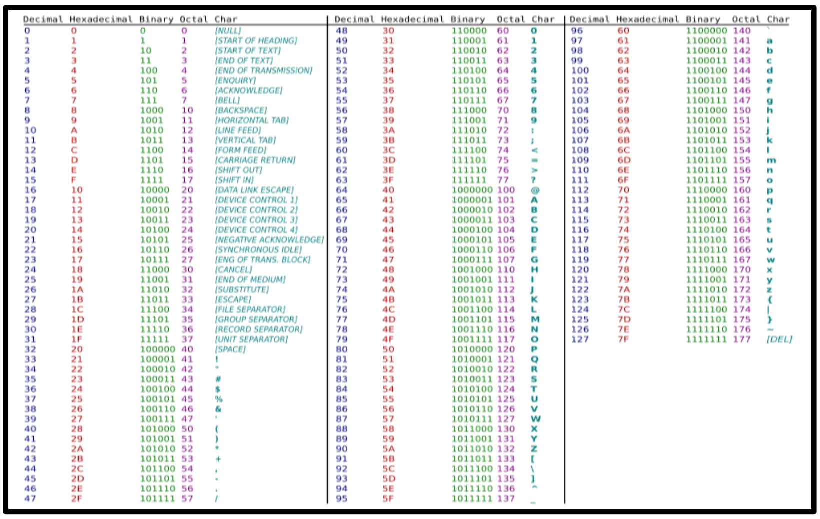


3.2.1 Character Encoding

Encoding standard/scheme refers to a specific set of rules and guidelines that define how information is represented and stored in an electronic format. **ASCII** and **UNICODE** are examples of encoding standards.

Difference between **ASCII** and **UNICODE**

Parameter	ASCII	UNICODE
Full form	American Standard Code for Information Interchange	Universal Character Set
Mutual relationship	ASCII is a subset of the UNICODE encoding scheme (UNICODE is a superset of ASCII) They share the first 0 to 127 bits.	
Usage	ASCII is used for electronic communication and in programming languages such as HTML.	Unicode encodes all the world's written characters into **one universal system** so that browsers everywhere can use only one encoding system. This also allows multiple languages on the same page to be displayed simultaneously.
Supporting characters	ASCII can represent only English alphabets, digits and standard special symbols. ● Numbers: 48-57 ● Upper case: 65-90 ● Lower case: 97-122	UNICODE can represent a large range of characters: e.g. special symbols, formulae, emoji and characters from different languageIDSs **(Up to 32 bits to represent the character, allowing much wider range of characters compared to ASCII which only uses 7 bits to represent characters)**
Bits per character	ASCII uses 7-bit encoding scheme	UNICODE uses mainly four character encoding schemes: ● UTF-7 (7-bit) ● UTF-8 (8-bit) ● UTF-16 (16-bit) ● UTF-32 (32-bit)
Memory Consumption	ASCII consumes less memory	UNICODE consumes more memory as compared to ASCII



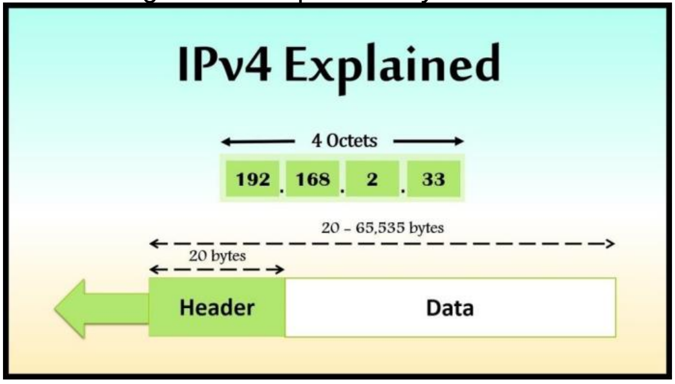
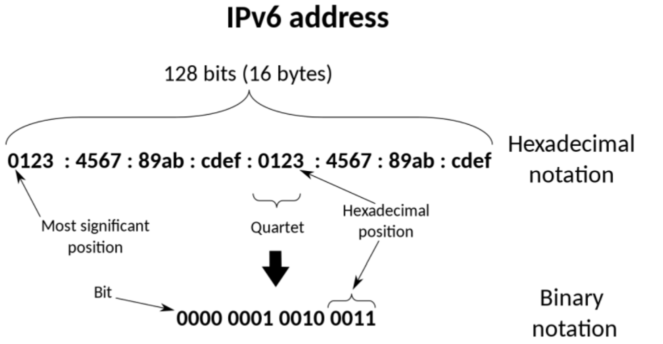
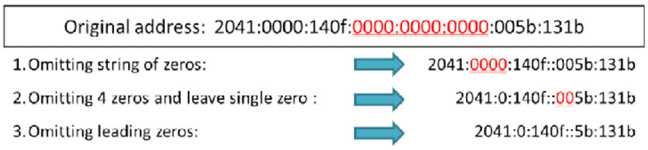
Decimal	Hexadecimal	Binary	Octal	Char	Decimal	Hexadecimal	Binary	Octal	Char	Decimal	Hexadecimal	Binary	Octal	Char
0	0	0	0	[NULL]	48	30	110000	60	0	96	60	1100000	140	`
1	1	1	1	[START OF HEADING]	49	31	110001	61	1	97	61	1100001	141	a
2	2	10	2	[START OF TEXT]	50	32	110010	62	2	98	62	1100010	142	b
3	3	11	3	[END OF TEXT]	51	33	110011	63	3	99	63	1100011	143	c
4	4	100	4	[END OF TRANSMISSION]	52	34	110100	64	4	100	64	1100100	144	d
5	5	101	5	[ENQUIRY]	53	35	110101	65	5	101	65	1100101	145	e
6	6	110	6	[ACKNOWLEDGE]	54	36	110110	66	6	102	66	1100110	146	f
7	7	111	7	[BELL]	55	37	110111	67	7	103	67	1100111	147	g
8	8	1000	10	[BACKSPACE]	56	38	111000	70	8	104	68	1101000	150	h
9	9	1001	11	[HORIZONTAL TAB]	57	39	111001	71	9	105	69	1101001	151	i
10	A	1010	12	[LINE FEED]	58	3A	111010	72	:	106	6A	1101010	152	j
11	B	1011	13	[VERTICAL TAB]	59	3B	111011	73	;	107	6B	1101011	153	k
12	C	1100	14	[FORM FEED]	60	3C	111100	74	<	108	6C	1101100	154	l
13	D	1101	15	[CARRIAGE RETURN]	61	3D	111101	75	=	109	6D	1101101	155	m
14	E	1110	16	[SHIFT OUT]	62	3E	111110	76	>	110	6E	1101110	156	n
15	F	1111	17	[SHIFT IN]	63	3F	111111	77	?	111	6F	1101111	157	o
16	10	10000	20	[DATA LINK ESCAPE]	64	40	1000000	100	@	112	70	1110000	160	p
17	11	10001	21	[DEVICE CONTROL 1]	65	41	1000001	101	A	113	71	1110001	161	q
18	12	10010	22	[DEVICE CONTROL 2]	66	42	1000010	102	B	114	72	1110010	162	r
19	13	10011	23	[DEVICE CONTROL 3]	67	43	1000011	103	C	115	73	1110011	163	s
20	14	10100	24	[DEVICE CONTROL 4]	68	44	1000100	104	D	116	74	1110100	164	t
21	15	10101	25	[NEGATIVE ACKNOWLEDGE]	69	45	1000101	105	E	117	75	1110101	165	u
22	16	10110	26	[SYNCHRONOUS IDLE]	70	46	1000110	106	F	118	76	1110110	166	v
23	17	10111	27	[ENG OF TRANS. BLOCK]	71	47	1000111	107	G	119	77	1110111	167	w
24	18	11000	30	[CANCEL]	72	48	1001000	110	H	120	78	1111000	170	x
25	19	11001	31	[END OF MEDIUM]	73	49	1001001	111	I	121	79	1111001	171	y
26	1A	11010	32	[SUBSTITUTE]	74	4A	1001010	112	J	122	7A	1111010	172	z
27	1B	11011	33	[ESCAPE]	75	4B	1001011	113	K	123	7B	1111011	173	{
28	1C	11100	34	[FILE SEPARATOR]	76	4C	1001100	114	L	124	7C	1111100	174	|
29	1D	11101	35	[GROUP SEPARATOR]	77	4D	1001101	115	M	125	7D	1111101	175	}
30	1E	11110	36	[RECORD SEPARATOR]	78	4E	1001110	116	N	126	7E	1111110	176	~
31	1F	11111	37	[UNIT SEPARATOR]	79	4F	1001111	117	O	127	7F	1111111	177	[DEL]
32	20	100000	40	[SPACE]	80	50	1010000	120	P					
33	21	100001	41	!	81	51	1010001	121	Q					
34	22	100010	42	"	82	52	1010010	122	R					
35	23	100011	43	#	83	53	1010011	123	S					
36	24	100100	44	$	84	54	1010100	124	T					
37	25	100101	45	%	85	55	1010101	125	U					
38	26	100110	46	&	86	56	1010110	126	V					
39	27	100111	47	'	87	57	1010111	127	W					
40	28	101000	50	(	88	58	1011000	130	X					
41	29	101001	51	)	89	59	1011001	131	Y					
42	2A	101010	52	*	90	5A	1011010	132	Z					
43	2B	101011	53	+	91	5B	1011011	133	[					
44	2C	101100	54	,	92	5C	1011100	134	\					
45	2D	101101	55	-	93	5D	1011101	135	]					
46	2E	101110	56	.	94	5E	1011110	136	^					
47	2F	101111	57	/	95	5F	1011111	137	_					

Other relevant applications

RGB Code is a model where primary colours are added together to form a wide range of colours.

Internet Protocol (IP) address is a numerical label assigned to each device connected to a computer network that uses the Internet Protocol for communication

Two kinds of IP addresses are used today:

IPv4 address	IPv6 address
● 32-bit address ● Supports 2^{32} number of addresses ● Usually presented as 4 denary numbers of range 0-255 separated by dots  IPv4 Explained 4 Octets 192 . 168 . 2 . 33 20 – 65,535 bytes 20 bytes Header Data ● Special IP addresses: ○ **127.0.0.1** (localhost) ○ **0.0.0.0** (all IPv4 address on local machine) ○ **255.255.255.255** (broadcast address)	● 128-bit address ● Supports 2^{128} number of addresses ● Usually presented as 8 groups of 4 hexadecimal digits separated by colons  IPv6 address 128 bits (16 bytes) 0123 : 4567 : 89ab : cdef : 0123 : 4567 : 89ab : cdef Hexadecimal notation Most significant position Quartet Hexadecimal position Bit 0000 0001 0010 0011 Binary notation ● For compactness, leading zeros and groups of 4 zeros are often omitted.  Original address: 2041:0000:140f:0000:0000:0000:005b:131b 1. Omitting string of zeros: → 2041:0000:140f::005b:131b 2. Omitting 4 zeros and leave single zero : → 2041:0:140f::005b:131b 3. Omitting leading zeros: → 2041:0:140f::5b:131b

Media Access Control (MAC) address is a unique address that is fixed for each device connected to a computer network.

- 48-bit address
- 6 groups of 2 hexadecimal digits separated by colons or hyphens.
- First 3 groups of 2 hexadecimal digits identify the manufacturer that produces the Network Interface Card (NIC).

Boolean algebra

NOT	AND	OR	XOR
NOT A = A’	A **AND** B = A · B	A **OR** B = A + B	A **XOR** B = A ^ B

Truth tables are visual representations of various inputs and their associated logical outputs.

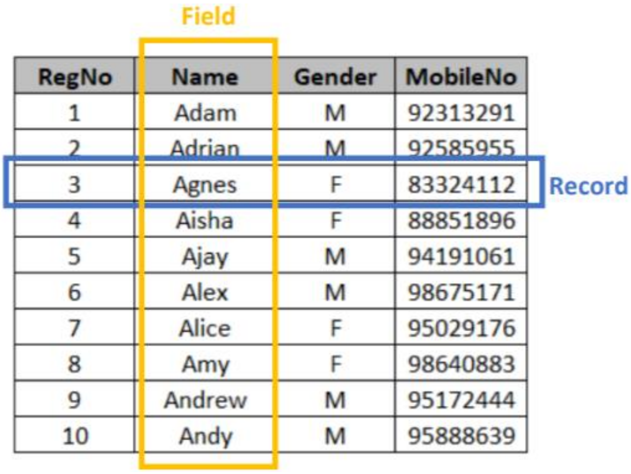
x	y	x · y	x + y
0	0	0	0
1	0	0	1
0	1	0	1
1	1	1	1

3.3 SQL Database

3.3.1 Relational Database Management System (RDBMS)

A **database** is a collection of data stored in an organised or logical manner.

Four basic functions of persistent storage: **Create, Read, Update, Delete** (CRUD).

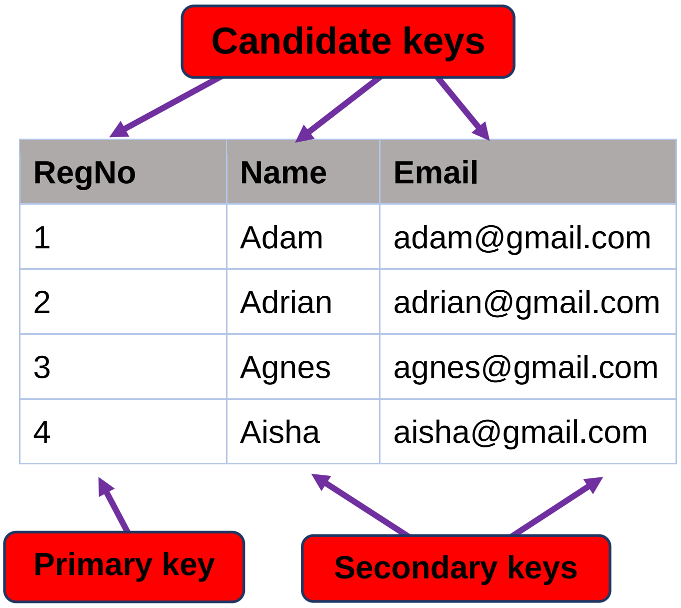

Field
Record

RegNo	Name	Gender	MobileNo
1	Adam	M	92313291
2	Adrian	M	92585955
3	Agnes	F	83324112
4	Aisha	F	88851896
5	Ajay	M	94191061
6	Alex	M	98675171
7	Alice	F	95029176
8	Amy	F	98640883
9	Andrew	M	95172444
10	Andy	M	95888639

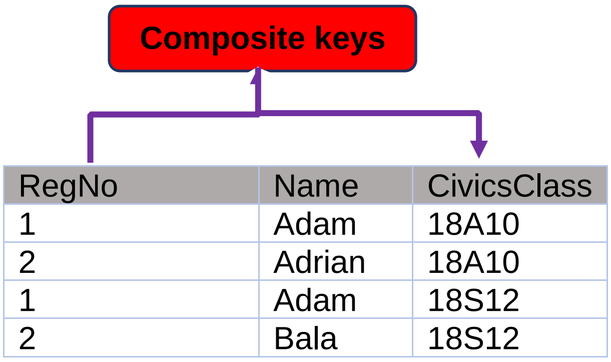
Table	A collection of record for a particular subject
Record (row)	A complete set of data about a single item
Field (column)	A set of values arranged in a table and has the same data type
Attributes	The describing characteristics or properties that define all items pertaining to a certain category applied to all cells of a column. "Quotation marks" needed for attribute names with a space.

3.3.2 Keys

Primary Key	A field in a database table that acts as a unique identifier for each record
Composite key	A combination of two or more fields in a table that can uniquely identify each record in a table.
Foreign key	A field in one table that refers to the primary key in another table, forming a relationship between the tables. **Qn: Purpose of foreign key + context** ● MatricNo in StudentCCA ensures that each CCA is associated with a valid student. ● CCAName in StudentCCA ensures that each student is associated with a valid CCA



RegNo	Name	Email
1	Adam	adam@gmail.com
2	Adrian	adrian@gmail.com
3	Agnes	agnes@gmail.com
4	Aisha	aisha@gmail.com



RegNo	Name	CivicsClass
1	Adam	18A10
2	Adrian	18A10
1	Adam	18S12
2	Bala	18S12

3.3.3 Data Redundancy, Data Inconsistency, Data Dependency

Data redundancy	There is unnecessary duplicate data found in multiple locations in the database. **Problems** 1. An unnecessary amount of data storage capacity required to store duplicate data 2. Increased risk of having inaccurate data hence, leading to data inconsistency 3. It is more time consuming and problematic to update the database because to maintain consistent data, all the duplicate data needs to be updated accurately. 4. Retrieval of data is slow and inefficient.
Data inconsistency	The same data exists in different formats in multiple tables as data is not updated correctly. This means the data lacks integrity.
Data dependency	**Functional dependency** ● A direct relationship where ***one attribute uniquely identifies another attribute*** in the same table. X -> Y (Y depends on X) **Transitive dependency** ● An indirect relationship formed by two functional dependencies. X -> Y -> Z

3.3.4 Normalisation

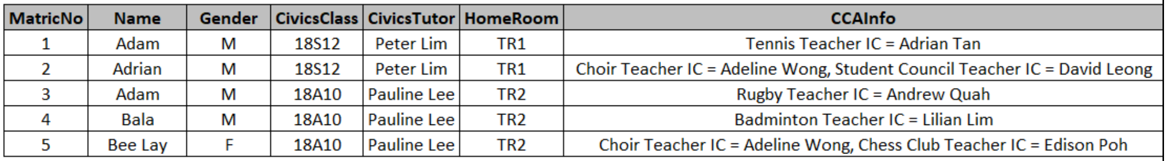
Normalisation

The process of organising the tables in a database to

1. reduce data redundancy so as to optimise storage space
2. prevent data inconsistency due to insert, update or delete anomalies

Unnormalized Form (UNF)

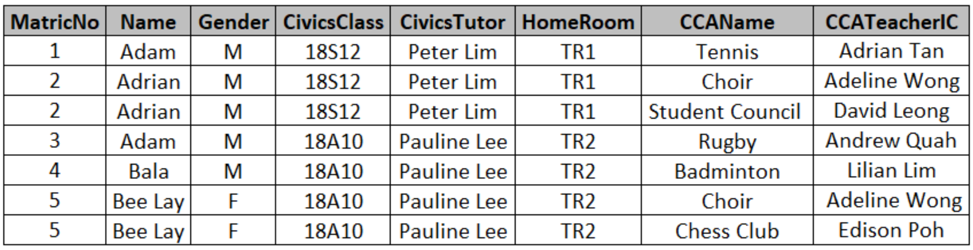
- No condition



MatricNo	Name	Gender	CivicsClass	CivicsTutor	HomeRoom	CCAInfo
1	Adam	M	18S12	Peter Lim	TR1	Tennis Teacher IC = Adrian Tan
2	Adrian	M	18S12	Peter Lim	TR1	Choir Teacher IC = Adeline Wong, Student Council Teacher IC = David Leong
3	Adam	M	18A10	Pauline Lee	TR2	Rugby Teacher IC = Andrew Quah
4	Bala	M	18A10	Pauline Lee	TR2	Badminton Teacher IC = Lilian Lim
5	Bee Lay	F	18A10	Pauline Lee	TR2	Choir Teacher IC = Adeline Wong, Chess Club Teacher IC = Edison Poh

First Normal Form (1NF)

- Each record has a primary key.
- Data in each field is atomic (a single value that cannot be sensibly sub-divided).
- Each record must have no repeating groups of attributes.

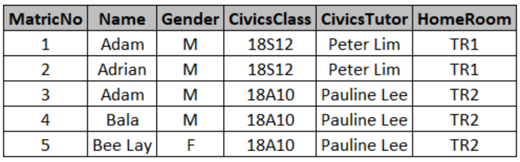


MatricNo	Name	Gender	CivicsClass	CivicsTutor	HomeRoom	CCAName	CCATeacherIC
1	Adam	M	18S12	Peter Lim	TR1	Tennis	Adrian Tan
2	Adrian	M	18S12	Peter Lim	TR1	Choir	Adeline Wong
2	Adrian	M	18S12	Peter Lim	TR1	Student Council	David Leong
3	Adam	M	18A10	Pauline Lee	TR2	Rugby	Andrew Quah
4	Bala	M	18A10	Pauline Lee	TR2	Badminton	Lilian Lim
5	Bee Lay	F	18A10	Pauline Lee	TR2	Choir	Adeline Wong
5	Bee Lay	F	18A10	Pauline Lee	TR2	Chess Club	Edison Poh

Second Normal Form (2NF)

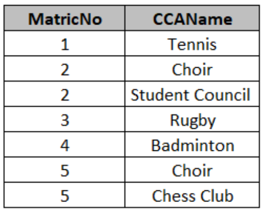
- In 1NF
- Every non-key attribute must be fully and functionally dependent on the primary key

Student



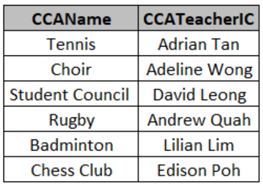
MatricNo	Name	Gender	CivicsClass	CivicsTutor	HomeRoom
1	Adam	M	18S12	Peter Lim	TR1
2	Adrian	M	18S12	Peter Lim	TR1
3	Adam	M	18A10	Pauline Lee	TR2
4	Bala	M	18A10	Pauline Lee	TR2
5	Bee Lay	F	18A10	Pauline Lee	TR2

StudentCCA



MatricNo	CCAName
1	Tennis
2	Choir
2	Student Council
3	Rugby
4	Badminton
5	Choir
5	Chess Club

CCAInfo

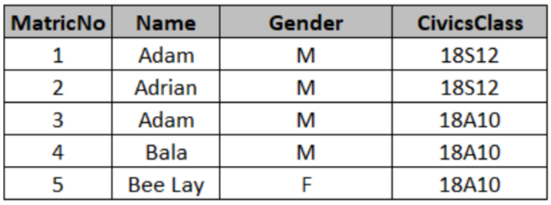


CCAName	CCATeacherIC
Tennis	Adrian Tan
Choir	Adeline Wong
Student Council	David Leong
Rugby	Andrew Quah
Badminton	Lilian Lim
Chess Club	Edison Poh

Third Normal Form (3NF)

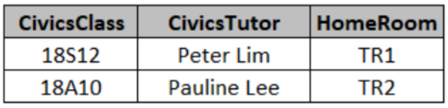
- In 2NF
- Table should not have transitive dependencies i.e. all fields must only be determined by primary/composite key, not by other attribute

Student



MatricNo	Name	Gender	CivicsClass
1	Adam	M	18S12
2	Adrian	M	18S12
3	Adam	M	18A10
4	Bala	M	18A10
5	Bee Lay	F	18A10

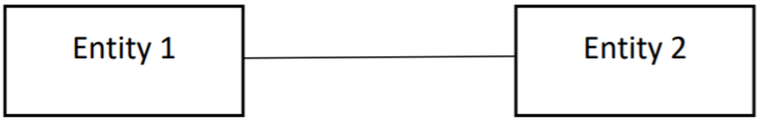
Civics



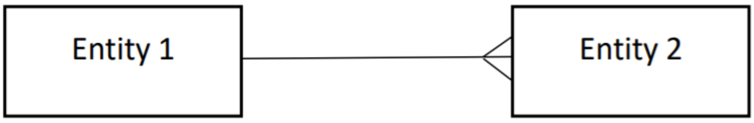
CivicsClass	CivicsTutor	HomeRoom
18S12	Peter Lim	TR1
18A10	Pauline Lee	TR2

3.3.5 Entity-Relationship (ER) Diagram

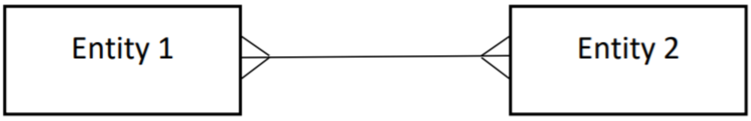
1. One to One



2. One to Many



3. Many to Many



Tip: Look out for words such as: **"each"**, **"unique"**, **"specific"**. They imply that the table is one-sided. Similarly, words like **"more than one"** imply that the table becomes many-sided.

Example: Database of students, their respective CCAs and CCA information

Legend:
___: Primary Key
: Foreign Key

Table description

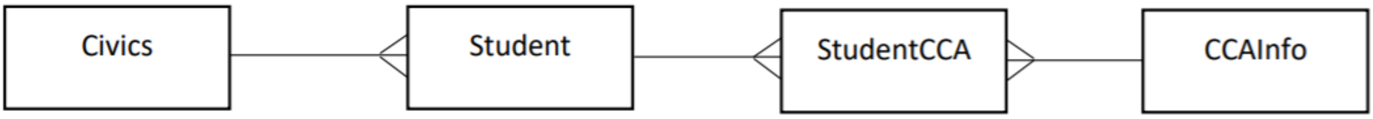
Student (MatricNo, Name, Gender, CivicsClass)

Civics (CivicsClass, CivicsTutor, HomeRoom)

StudentCCA (MatricNo, CCAName)

CCAInfo (CCAName, CCATeacherIC)

E-R diagram



3.3.8 SQL Query

Example
Person(PersonID, Name, Age)
Employee(EmployeeID, HoursWorked)
Company(**PersonID***, **EmployeeID***)

1. CREATE TABLE

```
CREATE TABLE 'Person' (
'PersonID' INTEGER PRIMARY KEY AUTOINCREMENT,
'Name' TEXT NOT NULL,
'Age' INTEGER NOT NULL CHECK(Age > 0 AND Age < 100)
);

CREATE TABLE 'Employee' (
'EmployeeID' INTEGER PRIMARY KEY AUTOINCREMENT,
'HoursWorked' REAL NOT NULL
);

CREATE TABLE 'Company' (
'PersonID' INTEGER NOT NULL,
'EmployeeID' INTEGER NOT NULL,
PRIMARY KEY('PersonID', 'EmployeeID'),
FOREIGN KEY('PersonID') REFERENCES Person('PersonID'),
FOREIGN KEY('EmployeeID') REFERENCES Employee('EmployeeID')
)
```

2. INSERT INTO

```
INSERT INTO Person('Name', 'Age')
VALUES(?, ?)
```

3. SELECT FROM

```
SELECT*
FROM Person
```

a. IS NOT NULL

```
WHERE Person.Age IS NOT NULL
```

b. ORDER BY

```
--Write an SQL query to output the number of bus services of each
bus operator along "Tampines Ave 8"

SELECT Operator, COUNT(DISTINCT ServiceNo) FROM Route
INNER JOIN Stop
ON Route.BusStopCode = Stop.BusStopCode
WHERE RoadName = 'Tampines Ave 8'
GROUP BY Operator;
```

c. GROUP BY

```
GROUP BY Person.Name
```

d. SUM, AVG, COUNT, MIN, MAX

```
SELECT SUM(Employee.HoursWorked)
FROM Employee
```

4. UPDATE SET

```
UPDATE Person
SET Name = 'Pablo', Age = 18
WHERE PersonID = 1
```

5. DELETE FROM

```
DELETE FROM Person
WHERE Person.PersonID = 1
```

6. DROP TABLE

```
DROP TABLE Company
```

Link to more SQL syntax: SQL Annex

3.3 NoSQL Database

3.3.6 SQL vs NoSQL

	SQL Database	NoSQL Database
Relational/ Non-relational	Relational Database Management System (RDBMS).	Non-relational Database Management System.
Terms	Database Table Record (Row) Field (Column)	Database Collection Document Field
Schema	SQL databases have a **fixed predefined schema** that prevents changes to be made easily. If a developer wishes to add a field to a small number of records, he needs to **include the field for the entire table**.	NoSQL databases have a **dynamic schema** that allows changes to be made easily. New fields can be added **without making changes to existing schema**. Thus, it handles unstructured data efficiently.
Data types of each field	Fixed	Flexible
Join & Complex queries	Supported	Not supported
Data storage	Tables are joined to each other via primary and foreign keys. SQL database does not support hierarchical data storage. Hence, the cost of storing data in a SQL database is **more expensive** than storing the same amount of data in a NoSQL database.	NoSQL supports **hierarchical data storage** where data is stored in a tree-like structure. The data at the root node will be most frequently accessed and likely to be stored in more expensive, faster storage devices. Less frequently used data is moved to cheaper slower storage devices.
Scalability	SQL databases are vertically scalable, meaning that improving its server usually requires upgrading an existing server with faster processors and more memory. Such high-performing components can be **expensive,** and upgrades are limited by the capacity of a single machine.	NoSQL databases are horizontally scalable, meaning that its performance can be improved by simply increasing the number of servers. This is **relatively cheaper** as mass-produced average-performance computers are easily available at low prices.
Impact of server failure on performance	SQL databases are stored in a single server, making it unavailable when the server fails.	NoSQL databases are designed to take the advantage of multiple servers so that if one server fails, the other servers can continue to support applications.

Performance speed	NoSQL databases generally have a better performance speed for handling read-heavy workloads than SQL databases. This would lead to less waiting time for scenarios such as online transactions.	
Data Integrity	Since SQL databases follow ACID transactions, they can ensure data integrity.	NoSQL databases, by design, do not emphasise traditional data integrity mechanisms like ACID transactions, making them vulnerable to data inconsistency. Hence, they may not necessarily ensure data integrity.

3.3.7 Applications of SQL and NoSQL

Factors to consider when choosing between SQL and NoSQL database

SQL Database	NoSQL Database
Structured data which requires fixed schema	Unstructured data with flexible data types which requires dynamic schema
Complex queries are frequently performed	Data storage needs to be performed quickly
High number of simultaneous transactions	Storing of extremely large amount of data (big data) is required
Large budget	Small limited budget
The transactions must conform to ACID properties to protect the integrity of the database	

ACID Properties

Atomicity	A change in the database must either be **performed completely or not performed at all**. The software must prevent a half-performed transaction from being saved.
Consistency	A transaction must take a whole database from one consistent state to another consistent state, for example in a bank transfer transaction the amount of money in the whole system must be the same at the end of the transaction as it was at the beginning.
Isolation	It is important that a transaction should be performed in isolation so that other users or processes cannot have access to the data concerned **until the new consistent state has been committed** (saved). In practice, this means that while an operation is being performed in a record, the record is locked. This may involve making the record invisible to others or it may only lock the record for writing. After the transaction is committed, the record may be unlocked again.
Durability	After a transaction successfully completes, changes to data **persist** and are **not undone**, even in the event of a server failure.

Link to more ACID properties: https://www.geeksforgeeks.org/acid-properties-in-dbms/

3.3 Data Protection

3.3.9 Data Privacy, Data Integrity

Data privacy	Data integrity	Data security
Requirement for data to be accessed by or disclosed to authorised people only	A concept and process that ensures the accuracy, completeness, consistency and validity of an organisation's data	The protection of data from unauthorised users who may have an interest in the data and an intention in corrupting it

3.3.10 Methods to Protect Data

Data Privacy	Data Integrity
Data encryption	Backup data.
End-point authentication methods	Validate input data.

3.3.11 Backup, Archive

	Backup	Archive
Definition	Creating copies of data and storing them separately as a safety precaution or prevention against data corruption or loss.	Storing **inactive** data that is kept for historical references or auditing purposes, ensuring that important records remain available years after they were created.
Purpose	The backup copies will allow the data to be restored to the latest backup point hence, reducing the amount of data loss and the business to recover and resume operation quickly after the event.	Some businesses are required by some regulatory organisations to archive the old data for a period of time such that it can be referred to again in the future when the need rises.
Data Storage Method	The original data remains in place, while a backup copy is stored in another location	Archived data is moved from its original location to an archive storage location
Data State	Backed-up data is constantly changing	Once you create an archive, you do not modify it
Data Retention Policy	You periodically delete or overwrite data backups that are too old to be useful	Data archives are designed for long-term storage
Storage Type	Hot cloud storage or easily accessible local storage locations	Cold cloud storage or tape archives

Data Scope	All of your data, with the exception of unimportant information like temporary files	Specific files that you must retain for compliance purposes

3.3.12 Version Control, Naming Convention

Version control	File Naming convention (FNC)
A system that records changes to a file or set of files over time so that you can recall specific versions later ● Allows users to rollback to previous working versions of processes to fix mistakes while minimising disruption to other team members. ● ***Promotes team collaboration*** E.g. **Git** ● Modified - You have changed the file but have not committed to your database yet ● Staged - You have marked a modified file in its current version to go into your next commit snapshot. ● Committed - The data is safely stored in your local database.	A framework for naming files in a way that describes what they contain and how they relate to other files. ● Minimise the chances of files being misplaced or lost unintentionally due to poor organisation of files. ● Quick location of files

3.3.13 Personal Data Protection Act (PDPA)

Personal Data Protection Act (PDPA) is a data protection law that governs the collection, use, disclosure and care of personal data by organisations.

	Personal Data Protection Act (PDPA)
Consent	Organisations must obtain an individual's knowledge and consent to collect, use or disclose personal data
Accountability	Organisations must make information about their personal data protection policies available on request. They should also make available the business contact information of the representatives responsible for answering questions relating to the organisations' collection, use, or disclosure of personal data.
Notification	Organisations must notify the individual of the purpose for collecting, using or disclosing their personal data.
Appropriateness	Organisations may collect, use or disclose personal data only for purposes that would be considered appropriate to a reasonable person under the given circumstances.

9 main personal data obligations

Obligations	Description	Network security feature
Consent obligation	Only collect, use or disclose personal data for purposes for which an individual has given his or her consent.	Data encryption
Purpose limitation obligation	An organisation may collect, use or disclose personal data about an individual for the purposes that a reasonable person would consider ***appropriate*** in the circumstances and for which the individual has ***given consent***.	File permission, Encryption
Notification obligation	Notify individuals of the purposes for which your organisation is intending to collect, use or disclose their personal data.	IPS, IDS to promptly detect and notify about security breaches. Automatic alerting systems for unusual network activity
Access and correction obligation	Upon request, organisations have to provide individuals with access to their personal data and information about how the data was used or disclosed within a year before the request.	Use IDS to maintain detailed logs of user access and activities to facilitate auditing and tracking changes
Accuracy obligation	Make a reasonable effort to ensure that personal data collected is accurate and complete, if it is likely to be used to make a decision that affects the individual or to be disclosed to another organisation.	Cryptographic hash functions and digital signature to ensure message integrity
Protection obligation	Make reasonable ***security arrangements*** to protect the personal data that the organisation possesses or controls to ***prevent unauthorised access, collection, use or disclosure***.	Firewall, IPS, IDS, Encryption, passwords, 2FA
Retention limitation obligation	Cease retention of personal data or dispose of it in a proper manner when it is no longer needed for any business or legal purpose.	Implement automated data deletion processes to ensure compliance with retention limitations
Transfer limitation obligation	Transfer personal data to another country only according to requirements prescribed under the regulations.	Secure protocols and encryption for secure data transfer
Accountability obligation	Make information about your data protection policies, practices and complaints process available on request	IDS to maintain audit trails and logs to track who accessed data and made changes.

Link to Network Protection Mechanisms

3.4 Ethics

3.4.1 Code of Conduct

	Code of Conduct
Professionalism	• Use their special knowledge and skill for the advancement of human welfare • not seek personal advantage to the detriment of the Society • Act with professionalism to enhance the prestige of profession/company
Integrity	• Not claim to a level of competence that one does not possess • Act with complete discretion when entrusted with confidential information • Give credit for work done by others where credit is due
Responsibility	• Adhere to their employers' or client's standards and guidelines • Indicate to their employers or client's consequences to be expected if their profession judgement is overruled.
Competence	• Continue to upgrade their knowledge and skills, and be aware of relevant development in the technology they are involved in • Extend public knowledge, understanding and appreciation of information technology and to oppose false or deceptive statements related to information technology of which they are aware

3.4.2 Impact of Computing

Social Impact

- Enables long distance communication
- Provides a voice to the underprivileged

Economic Impact

- ↑quality and quantity of G&S
- ↓COP

Social Issues

- Racist AI
- Gaming addiction

Economic Issues

- ↑technology (AI) → ↑workers without relevant skills → ↑structural UnE
- Dark web → rise of black market

Ethical Issues

- Invasion of privacy
- Violation of intellectual property rights

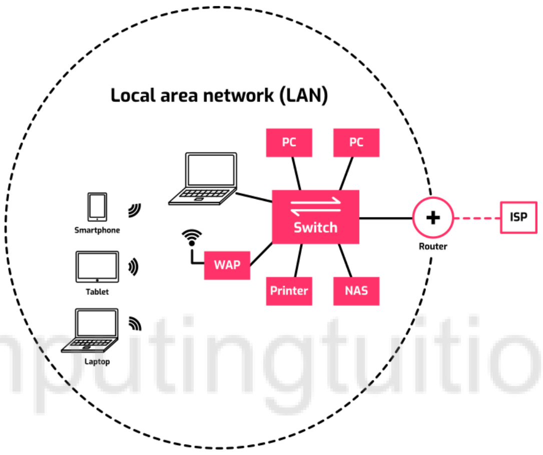
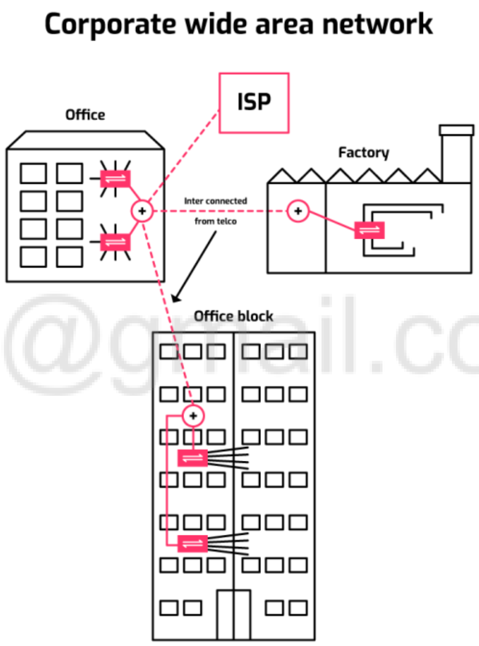
Legal Issues

- Black hat hacking
- Misuse of personal data

4.1 Computer Networks

4.1.1 Local Area Network (LAN), Wide Area Network (WAN), Intranet, Internet

Computer network is a group of computers that use the same set of communication protocols to share resources over digital connections.

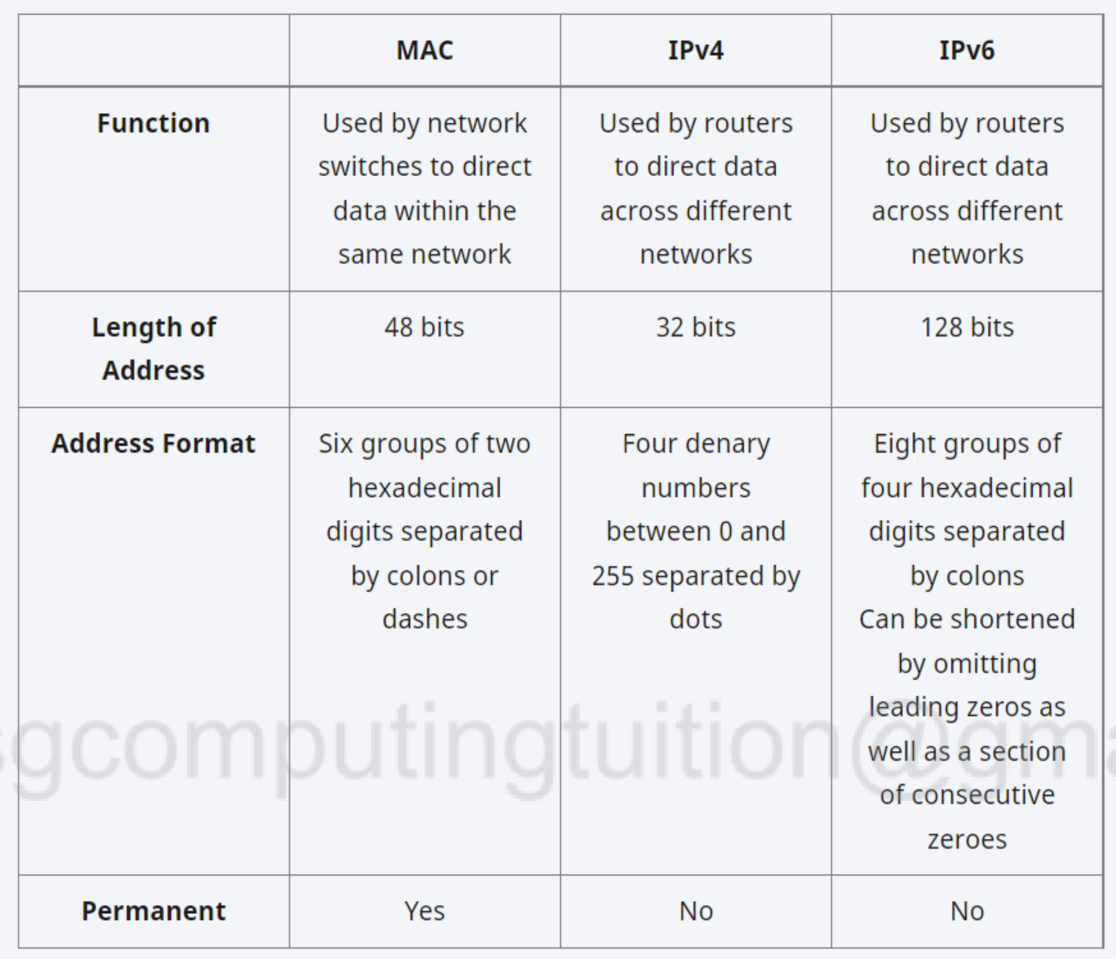
	Local Area Network (LAN)	Wide Area Network (WAN)
Definition	LAN is a collection of devices and computers connected together in one physical location of a small geographic area such as a building, office or home. E.g. Wifi, Ethernet	A collection of LANs and other networks that connect devices from multiple locations across the globe, covering a large geographic area. E.g. Internet
Visualisation	 Local area network (LAN) PC PC Switch Router ISP WAP Printer NAS Smartphone Tablet Laptop	 Corporate wide area network ISP Office Factory Inter connected from telco Office block

Intranet	Extranet	Internet
Intranet is a local private network within an organisation which **excludes outsider access**. It is a restricted network mainly used to share files and information securely within an organisation. Intranets are typically password-protected and **can only be accessed from within the organisation's premises** or **through a secure Virtual Private Network (VPN)**.	Extranets are a type of private network that allows approved external users, such as key customers, suppliers and partners, to access certain parts of an organisation's intranet. Extranets provide a secure platform for sharing information and collaborating with external parties without compromising the confidentiality of internal data.	Internet is the global system of interconnected computer networks that uses the **internet protocol suite** (TCP/IP) to communicate between networks and devices. Its network of publicly available web pages is called **World Wide Web (WWW)**.

4.1.2 Internet Protocol (IP) Addressing, Domain Name System (DNS) Server

Internet Protocol address (IP address) is a numerical label assigned to each device connected to a computer network that uses the Internet Protocol for communication.

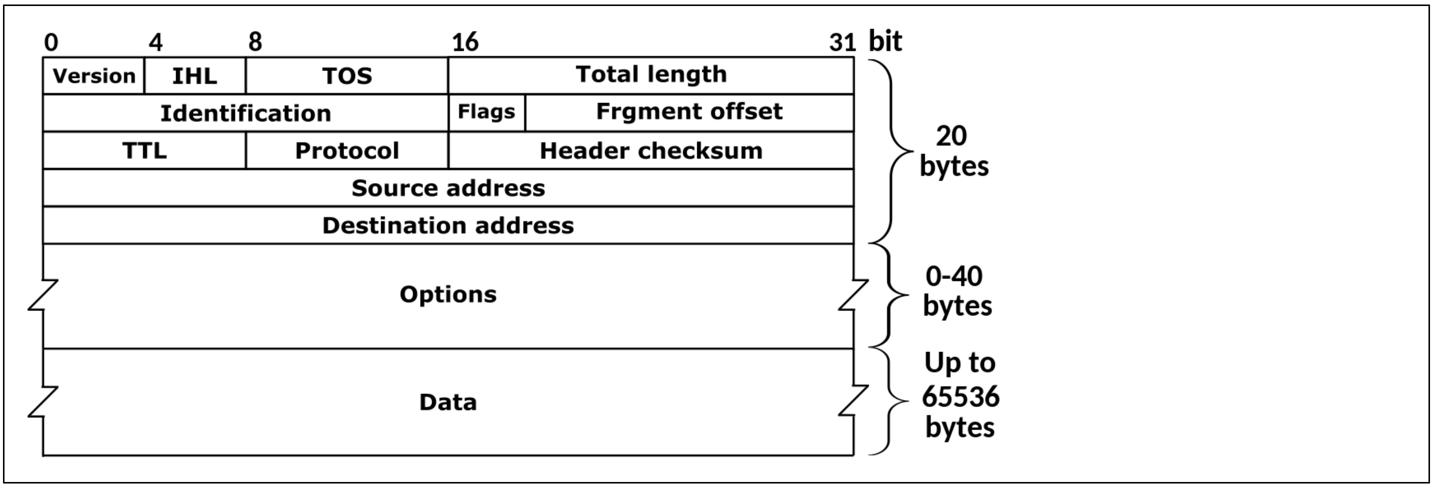
Explanations on IPV4 & IPV6 & MAC: Other relevant applications



	MAC	IPv4	IPv6
Function	Used by network switches to direct data within the same network	Used by routers to direct data across different networks	Used by routers to direct data across different networks
Length of Address	48 bits	32 bits	128 bits
Address Format	Six groups of two hexadecimal digits separated by colons or dashes	Four denary numbers between 0 and 255 separated by dots	Eight groups of four hexadecimal digits separated by colons Can be shortened by omitting leading zeros as well as a section of consecutive zeroes
Permanent	Yes	No	No

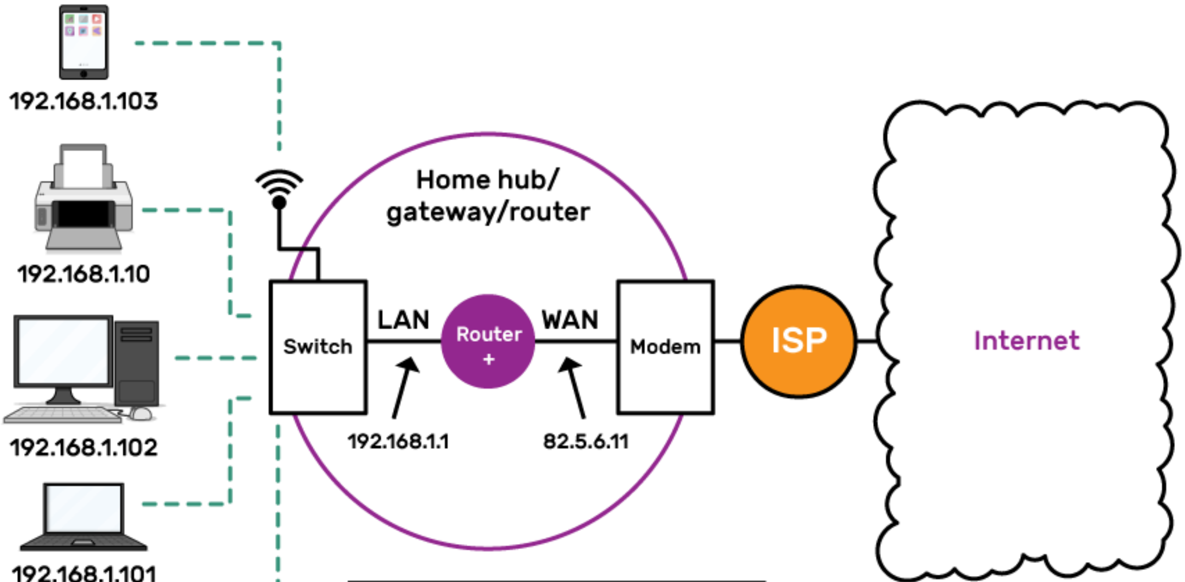
IP packets, routers and routing

Structure of IP Packet



Packet header includes essential details about the packet such as source and destination addresses, protocol, checksum and Time-To-Live to facilitate the transmission and processing within a network.

Diagram of a Home Network



Net	Int
192.168.1.0	LAN
82.0.0.0	WAN
0.0.0.0	WAN

E.g. The internal LAN's address is 192.168.1.1 while the ISP's WAN address is 82.5.6.11. Hence, the router is connected to two different IP networks which are automatically placed into the routing table. *(Router acts like a departure/immigration officers at the airport)*

How routers route IP packets across different networks.

- It compares the **IP address of destination in the packet header** and the **routing table** to determine which connection (interface) to send the packet to.
- The router uses a **subnet mask** to divide the IP address into network and host portions to determine which subnet the destination belongs to and route the packet to the respective network switch.
- The network switch will refer to the **MAC address table** to direct the packet to the correct destination device.

Error detection (for packets)

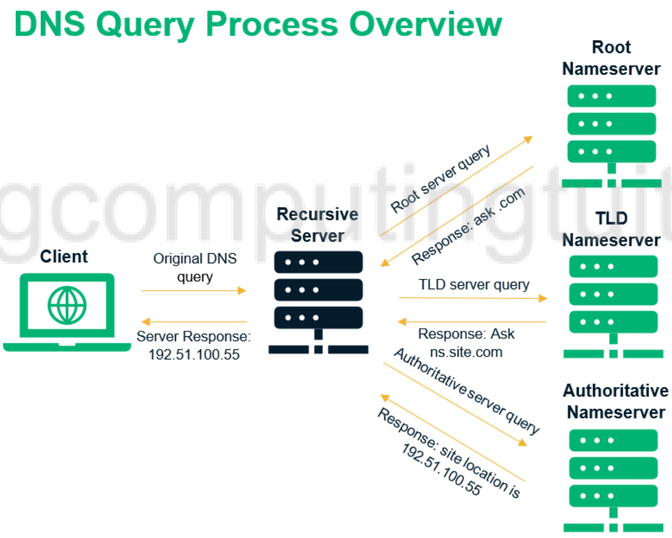
Checksum	Parity check
A small sized data derived from the dataset that is used to **detect and address errors that may occur during data transmission** to ensure the data received is the same as the data sent, not corrupted. The checksum algorithm works on the data to produce a checksum. This is appended to the packet header. Upon receiving, the receiver recalculates the checksum from the data received using the same algorithm and compares it with the one in the header. If the values are not the same, it is an indication that error must have occurred during data transmission.	Error-correction process in network communication that ensures data transmissions between communication nodes are accurate. The receiver agrees to use the same even parity bit or odd parity bit scheme as the sender. In odd parity, all packets sent have an odd number of 1s and in even parity, all packets sent have an even number of 1s. Odd parity is assumed for example here. Before the transmission, the parity bit is added to the original data to make it 'odd'. If the number of received bits is not odd, there has been an error in data transmission.

Uniform Resource Locator (URL) is a complete web address used to find a particular web page. The host's domain name is part of this URL.

Protocol	Third-level domain	Second-level domain	Top-level domain	Directory name	File name
http://	www	.example	.com	/directory	/file.txt

Types of DNS Services

Domain Name Server (DNS)			
A **decentralised** and **hierarchical** database (in structure) that converts URLs to their associated IP addresses			
Recursive DNS	**Root name server**	**Top-Level Domain (TLD) name server**	**Authoritative name server**
Service that takes URL requests from users and checks the records obtained from authoritative DNS servers for the associated IP address.	Root name servers are the servers at the root of the DNS hierarchy. Its primary function is to make redirection to the respective TLD server, given request. e.g. .com,.edu.gov	It makes a redirect to the respective SLD.	Service that answers DNS queries by converting domain names into IP addresses


DNS Query Process Overview
Root Nameserver
TLD Nameserver
Authoritative Nameserver
Recursive Server
Client
Original DNS query
Server Response: 192.51.100.55
Root server query
Response: ask .com
TLD server query
Response: Ask ns.site.com
Authoritative server query
Response: site location is 192.51.100.55

DNS query process

1. Client opens the web browser and types the URL in the address bar. E.g. example.com
2. **URL request** is sent to a ***recursive server***. Recursive server checks whether the domain name is found in its ***local cache***. If yes, go to step 7. Otherwise, go to step 3.
3. Recursive server sends a DNS query to a ***DNS root name server*** which then returns the address of Top Level Domain (TLD) E.g. .com
4. Recursive server sends the query to a ***TLD name server*** which then returns the address of domain's nameserver, example.com.
5. Recursive server sends the query again to an ***authoritative name server to map the domain names with the corresponding IP address***.
6. Authoritative name server sends the IP address back to the recursive server.
7. Recursive server sends the ***IP address of the host's web server*** back to the client and the client would be able to access the website.
8. Recursive server also caches the IP address for the domain.

4.1.3 Communication Protocols

Communication protocols are a set of rules for data transmission which are agreed upon by both sender and receiver.

Purpose of communication protocols

- Allow different networking devices to communicate with each other.
- Allow communication to be initiated and terminated.
- Configures a set of rules regarding the start and end of communication, the format and standard for transmission and how to handle transmission errors.

How TCP/IP layers help to fulfil the purpose of TCP/IP model:

- This process is complex because it involves how data should be sent/received (i.e packetized, addressed, transmitted, routed)
- This complexity is managed by ***breaking down its components into layers for specialisation/efficiency***. Each layer provides ***functionality that depends solely on the layer directly below it***
- ***Each layer works independently of one another****, allowing developers to modularise the processes in each layer. Should any error occur in the networking process, it would be* ***easy to identify defective codes*** *as it can be* ***traced by layers****.*
- *Developers just need to make sure that the data is in the* ***correct format*** *(so that it can move from one layer to the next*

OSI Model	TCP/IP Model	Protocol Data Unit	Network Devices	TCP/IP Protocol Suite	Function
Application Layer	Application Layer	Data	-	HTTP, SMTP, Telenet, FTP, DNS, RIP, SNMP	● ***Provides user interface, data encoding and data translation with help of protocols like DNS and HTTP.*** ● Transmission, interpretation and response of HTTP/SMTP/FTP request ● Produce data to be transmitted. ● Displays data to user
Presentation Layer			-		● Extracts data from application layer ● Translation (human-readable text to binary numbers) ● Data compression (to reduce number of bits transmitted)
Session Layer			-		● Establishes connections, maintains and terminates sessions ● Responsible for authentication, authorisation and security
Transport Layer	Transport Layer	Segments	-	TCP, UDP	● ***Provides end-to-end connection establishment of data segments and delivery with error control by using TCP and UDP protocols.*** ● Attaches port number or see which port it needs to go into ● Provides acknowledgement of successful

					data transmission ● Re-transmits data if error is found (using checksum) ● Part of the OS
Network Layer	Internet Layer	Packets	Router	ARP, IP, IGMP, ICMP	● ***Provisions the logical addressing and routing of data packets using the internet protocol like IPV4.*** ● Transmission of data between hosts over different networks ● Packet routing ● Places sender and receiver's IP addresses in the header
Data Link Layer	Network Access Layer	Frames	Switch, Bridge, Network Interface Card	Ethernet, Token RIng, ATM, Frame Relay	● ***Combines raw data into data frames, provides the physical interface for data transmission using protocols like PPP and ethernet.*** ● Ensures data transfer over nodes are error-free ● Handled by the Network Interface Card (NIC) and device drivers of host devices ● Attaches MAC addresses
Physical Layer		Bits	Modem		● Converts the bits into signals which are transmitted through optical fibres

4.1.4 Packet Switching

Packet switching

- The data transmitted is **too big** to fit in one packet.
- Data is divided into **packets**, each with its own ID number.
- Each packet is routed from one node to another on the internet from source to the destination.
- At each point, each node decides where to forward the packet based on **network traffic** and **node availability** using the **optimal paths** and sharing the link capacity (bandwidth) on a **packet-by-packet basis**.
  - Makes efficient use of **transmission capacity**.
  - Improves the overall **transmission speed**.
  - Prevents network **congestion**.
- **Re-transmission** is only required for the packets which are **lost or corrupted** during transmission. This prevents the need for sending the whole data again.

Sequential numbering of packets

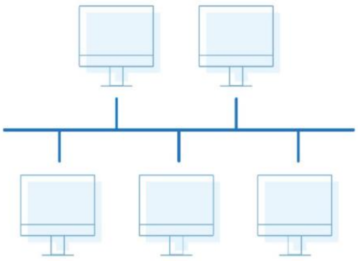
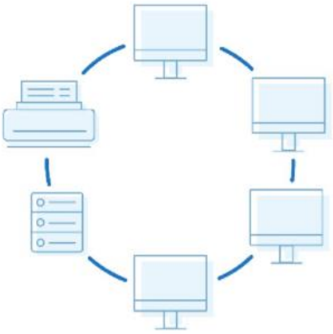
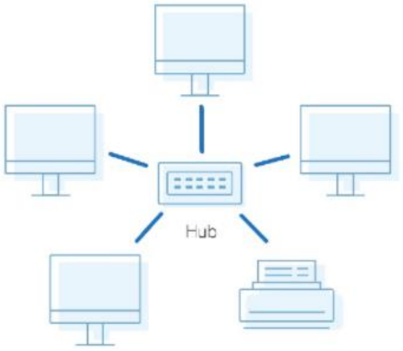
- Since the packets travel in different paths from sender to the receiver, they arrive in a different order.
- Hence, they need to be **sequentially numbered before transmission** so that they can be **assembled back into the original order** at the receiving end.
- Without sequence numbers, assembling the packets in order of arrival can lead to an incorrect sequence.
- Packets are **buffered**, waiting for **out-of-order packets** and **re-arrangement** of packets.
- The problem can be minimised with **additional data compression** which speeds up transmission time.

	Circuit Switching	Packet Switching
Definition	Process of setting up a **dedicated** communications channel before information can be transmitted.	Process of breaking up data into packets before information can be transmitted. Packets are individually routed from the source to the destination.
Dedicated Path	Circuit switching requires a dedicated path for communication.	Packet switching does not require a dedicated path for communication.
Path	Path is established for the entire conversation.	Path is established for each packet.
Bandwidth	Circuit switching has fixed bandwidth.	Packet switching has dynamic bandwidth as it can be **shared for transmission.**
Delay	Circuit switching has call setup delay.	Packet switching has packet transmission delay.

4.1.5 Organisation & Topology

Organisation	Client-Server Architecture	Peer-to-Peer (P2P) Network
Definition & Explanation	Computing model where an ***always-on host***, called server, **services requests** from several hosts, called clients. Client and server have ***distinct tasks*** E.g. When a web server ***receives a request*** for an object from a client host, it ***responds by sending the requested object to the client host***. The clients (web browsers) ***do not directly communicate with each other***.	Computing model where all hosts ***provide*** and ***request*** for the service, ***distributing the load among all computers***. It exploits direct communication between pairs of intermittently connected hosts, called peers.
Management	Ease of management due to centralised control of data and resources. ● Resources are updated faster ● Easier to perform backup	Poor management due to lack of central control of data and resources
Security	High security due to use of specialised security software for servers which monitors and protects resources on the network.	Low security due to lack of central management of access rights
Bandwidth	Stable and typically high speed but restricted by the number of clients per server	May achieve high speeds depending on the number of peers connected. Unstable bandwidth as not all peers would be online.
Cost	High of cost of set-up and maintenance	Cheaper to set up
Reliability	Client-Server Architecture is unreliable as the failure of the server leads to failure of the client.	P2P Network is reliable as the failure of one peer does not lead to failure of other peers.

(Only for O-level syllabus)

Topology	**Bus topology**	**Ring topology**	**Star topology**
Definition & Explanation	A common cable or backbone known as the bus connects all the devices. All the devices connected to the bus can detect data that is being transmitted, but only the intended recipient will accept and process the data.	Each network device is connected in adjacent to two other network devices. Data is passed around in a single direction until it reaches the intended recipient. If a cable or network device fails, the network fails.	A central network device connects to all other network devices. The central network device forwards data from the sender to the intended recipient.
Diagram			
Cost	Easy and cheap to install due to less cabling	Cheap to install	Higher costs due to more cabling, expensive central device
Installation (scalability)	Scalable as more network devices can be added simply.	Network has to be disrupted in order to add a new network device.	Scalable as network devices only need to be connected to the central network device.
Efficiency	Time inefficient		Time efficient
Reliability	Bus topology is unreliable as a break along the bus may disable part of the network. However, even when one network device breaks down, the network continues to operate.	Unreliable as the failure of one device/ break along the ring leads to failure of the network.	Most reliable as the network can continue to operate even when one network device is disrupted. The faulty network device can also be easily traced and replaced without affecting the whole network. However, if the central network device fails, the whole network stops working.
Others	Limited capacity as performance slows down with more network devices	Able to operate over larger distances and handle more data over one-way traffic. Central server not required to manage traffic	The load on each cable is reduced as each network device has its own dedicated cable.

4.2 Web Application

4.2.1 Web Application, Native Application

Feature	Web Application	Native Application
Definition	Application that can be accessed by a web browser.	Application that can only be accessed by a particular operating system.
Platform	Developed to be accessed via the device's internet browser. This allows for greater **compatibility** on different platforms	Developed for a specific operating system or platform.
Installation	No need to be downloaded or installed, allowing it to be accessed from any device.	Need to be installed in the device to function.
Update	Updated automatically as deployment & maintenance are done on a single set of server machines.	Incur high maintenance and update costs as deployment and any maintenance has to be done on individual client machines separately.
System Resources	Limited access to system resources such as GPS, camera	Have access to system resources such as GPS, camera
Internet Connection	Need an active internet connection to work	Usually able to work offline (except for updates)
Speed	Relatively slower, depending on the internet connection	Relatively faster
Security	Relatively less secure as they are inherently designed for high accessibility	Relatively safer and more secure as native applications have better authorization and administrators have better control

4.2.2 Usability Principles

Link to UI/UX Principle Notes: Jacob's Heuristic

Principle	Explanation	Example
Visibility of System Status	The design should always keep users informed about what is going on, through appropriate feedback within a reasonable amount of time	● "You are Here" maps ● Checkout flow in e-commerce websites
Match between System and the Real World	The design should speak the user's language. Use words, phrases, and concepts familiar to the user, rather than internal jargon. Follow real-world conventions, making information appear in a natural logical order.	● Shopping cart icon ● Grabbing hand icon
User control and freedom	Users often perform actions by mistake. They need a clearly marked "emergency exit" to leave the unwanted action without having to go through an extended process.	● Undo & Redo ● Cancel button ● Forget password
Recognition rather than recall	Minimise the user's memory load by making elements, actions and options visible. The user should not have to remember information from one part of the interface to another. Information required to use the design should be visible or easily retrievable when needed.	● Comparison table

4.3 Network Security

4.3.1 Information Security Threats

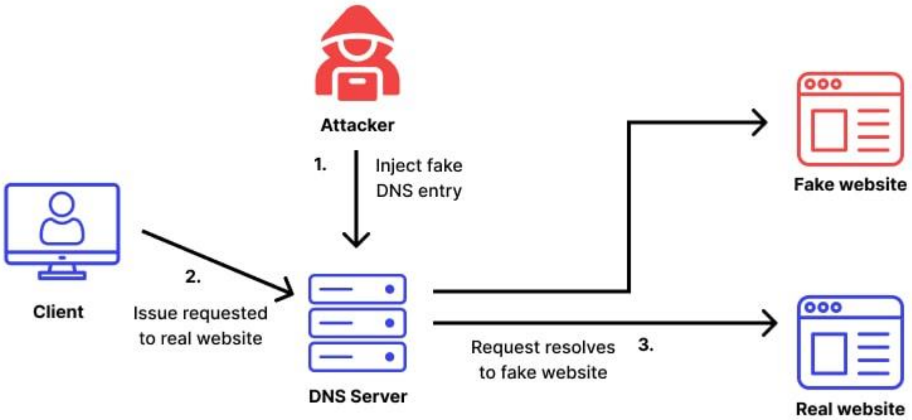
Network security is to secure the infrastructure of privately-owned networks/internet.

Elements of Secure Network	
Confidentiality	Preventing unauthorised persons from gaining access to unauthorised information.
Integrity	Preventing attackers from modifying data.
Availability	Making sure that resources are available for authorised users.

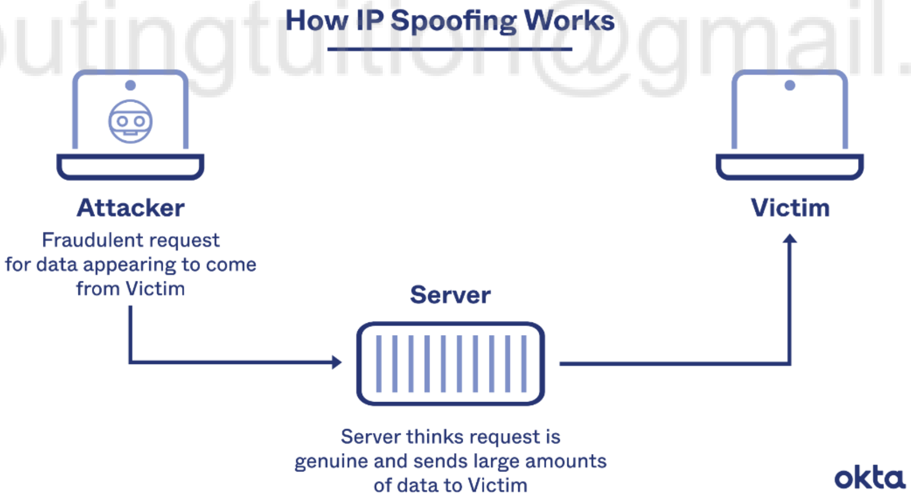
Cyberattack is an intentional attempt to damage, disrupt or gain unauthorised access to one or more computer systems through the internet.

<table>
<tr><th>Cyber-attack</th><th>Explanation</th></tr>
<tr><td><b>Denial of Service (DoS)</b></td><td>DOS attack sends a massive request to <b>exhaust server resources and bandwidth</b>, rendering the server, host, or other pieces of infrastructure unusable by legitimate users.

<b>Methods:</b>
<table>
<tr><td><b>Vulnerability attack</b></td><td>This involves sending <b><i>a few well-crafted messages</i></b> to a vulnerable application or operating system running on a targeted host. If the right sequence of packets is sent to a <b><i>vulnerable application or operating system</i></b>, the service can stop or, worse, the host can crash.</td></tr>
<tr><td><b>Bandwidth flooding</b></td><td>Sends a <b><i>deluge of packets</i></b> to the host—so many packets that the target's access link becomes clogged, preventing legitimate packets from reaching the server.</td></tr>
<tr><td><b>Connection flooding</b></td><td>The attacker <b><i>establishes a large number of half-open or fully open TCP connections</i></b> at the target host. The host can become so bogged down with these bogus connections that it stops accepting legitimate connections</td></tr>
</table></td></tr>
<tr><td><b>Phishing</b></td><td>The use of emails and fake websites that appear to be from reputable companies in order to steal personal information (E.g. credit card numbers, passwords)</td></tr>
<tr><td><b>Pharming</b></td><td>The interception of requests sent from a computer to a legitimate website and redirection to a fake website to steal personal data or credit card details. It is more difficult to detect than phishing as the fake website uses the same addresses as the real website.</td></tr>
</table>

<table>
<tr><td></td><td>

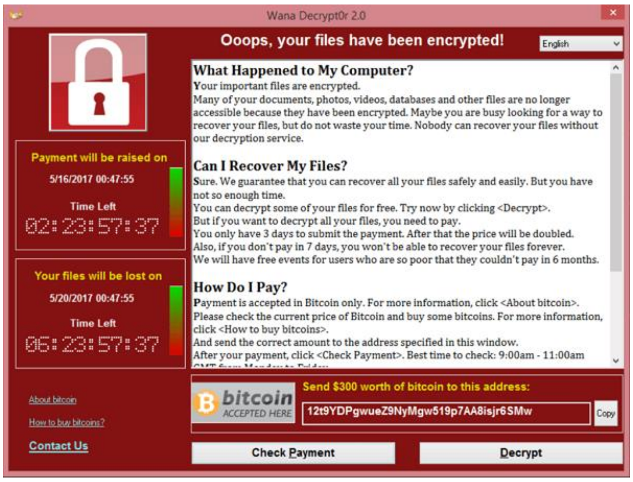
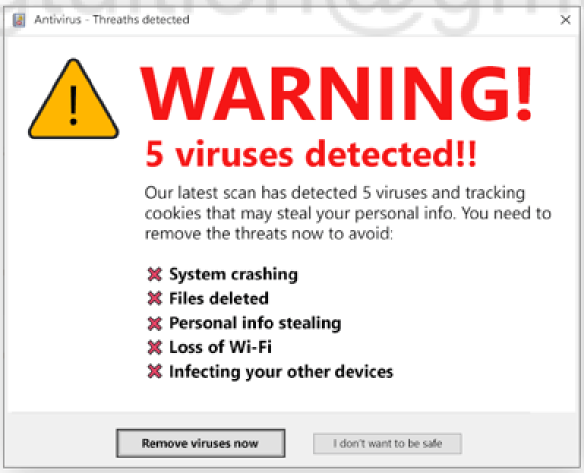
Attacker
1.
Inject fake DNS entry
Fake website
Client
2.
Issue requested to real website
DNS Server
Request resolves to fake website
3.
Real website

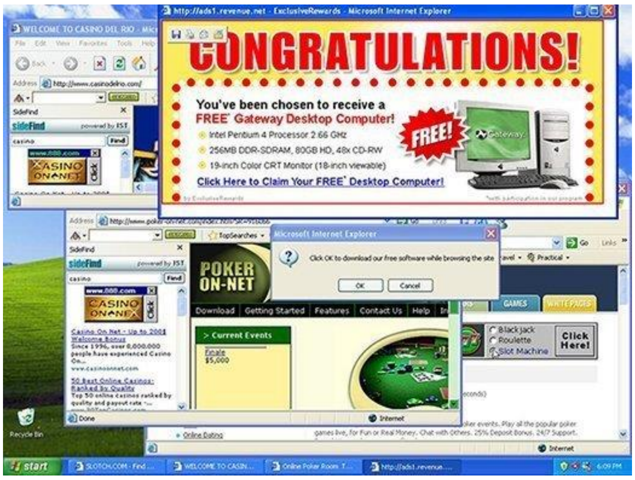
</td></tr>
<tr><td><b>Spamming</b></td><td>Mass distribution of unwanted messages or advertising sent to email addresses collected from sources such as public mailing lists, social networking sites such as public mailing lists, social networking sites, company websites and blogs. Emails are usually sent to users to lure them to enter their personal information and steal personal data.</td></tr>
<tr><td><b>Internet Protocol (IP) Spoofing</b></td><td>A type of cyber-attack that aims to trick networks that it is a legitimate entity by modifying the source address in the IP packet header.<ul><li>Server dutifully forward the packet to the destination</li><li>Unsuspecting victim performs some command embedded in the packet's content.</li></ul>

How IP Spoofing Works
Attacker
Fraudulent request for data appearing to come from Victim
Victim
Server
Server thinks request is genuine and sends large amounts of data to Victim
okta

</td></tr>
<tr><td><b>Packet Sniffing</b></td><td>A type of cyber-attack that aims to steal data by capturing data packets that are transmitted through a network.<ul><li>A passive receiver is placed in the vicinity of the wireless transmitter so that receiver can obtain a copy of every packet that is transmitted.</li><li>Usually targeted at the institution's access router to make a copy of every packet going to/from the organisation.</li></ul></td></tr>
</table>

Malware is a software that is intentionally used to damage, disrupt or gain unauthorised access to a computer system.

Malware	Explanation
Ransomware	A type of malware that blocks access to the victim's computer system until a certain amount of money, usually in bitcoin or other cryptocurrency is paid.  Wana Decrypt0r 2.0 Ooops, your files have been encrypted! English **What Happened to My Computer?** Your important files are encrypted. Many of your documents, photos, videos, databases and other files are no longer accessible because they have been encrypted. Maybe you are busy looking for a way to recover your files, but do not waste your time. Nobody can recover your files without our decryption service. **Can I Recover My Files?** Sure. We guarantee that you can recover all your files safely and easily. But you have not so enough time. You can decrypt some of your files for free. Try now by clicking <Decrypt>. But if you want to decrypt all your files, you need to pay. You only have 3 days to submit the payment. After that the price will be doubled. Also, if you don't pay in 7 days, you won't be able to recover your files forever. We will have free events for users who are so poor that they couldn't pay in 6 months. **How Do I Pay?** Payment is accepted in Bitcoin only. For more information, click <About bitcoin>. Please check the current price of Bitcoin and buy some bitcoins. For more information, click <How to buy bitcoins>. And send the correct amount to the address specified in this window. After your payment, click <Check Payment>. Best time to check: 9:00am - 11:00am Payment will be raised on 5/16/2017 00:47:55 Time Left 02:23:57:37 Your files will be lost on 5/20/2017 00:47:55 Time Left 06:23:57:37 About bitcoin How to buy bitcoins? Contact Us bitcoin ACCEPTED HERE Send $300 worth of bitcoin to this address: 12t9YDPgwueZ9NyMgw519p7AA8isjr6SMw Copy Check Payment Decrypt
Scareware	It is a program that attempts to frighten the victim into buying unnecessary software or providing their financial data. ● Scareware pops up on a user's desktop with flashing images or loud alarms, announcing that the computer has been infected. ● It usually urges the victim to quickly enter their credit card data and download a fake antivirus program.  Antivirus - Threaths detected WARNING! 5 viruses detected!! Our latest scan has detected 5 viruses and tracking cookies that may steal your personal info. You need to remove the threats now to avoid: ✖ System crashing ✖ Files deleted ✖ Personal info stealing ✖ Loss of Wi-Fi ✖ Infecting your other devices Remove viruses now I don't want to be safe
Adware	Software that automatically displays or downloads advertising material such as banners or pop-ups when a user is online.

	 CONGRATULATIONS! You've been chosen to receive a FREE* Gateway Desktop Computer! Intel Pentium 4 Processor 2.66 GHz 256MB DDR-SDRAM, 80GB HD, 48x CD-RW 19-inch Color CRT Monitor (18-inch viewable) Click Here to Claim Your FREE* Desktop Computer! Microsoft Internet Explorer: Click OK to download our free software while browsing the site [OK] [Cancel]
Spyware	A hidden program that secretly collects personal information about users and sends the information to attackers without the user's knowledge. Spywares record: ● Browsing history ● Keystrokes (specific to **Keylogger**) ● Credit card numbers
Fileless malware	● Fileless malware operates in the computer's memory and may evade detection by hiding in a trusted utility. ● Unlike traditional malware, fileless malware does not download code onto a computer, so there is no malware signature for a virus scanner to detect.
Cookies	● A small piece of data used by websites to store personal information on a user's web browser. ● Usually not malicious in nature, but sometimes used to collect personal information about others.
Trojan horse	● A computer program that pretends to be a harmless file or useful information **by hiding malicious codes**. ● Once a computer program runs, it harms computers by creating a backdoor access to intruders, allowing them to access and control the system.
Virus	● A malicious executable code that **attaches itself** to a normally harmless executable file and **modifies or deletes** it. ● When the modified programs are run by a user, the virus **attaches copies of itself** to any other programs it can find, thus **infecting** it.
Worm	● A standalone malware that **runs automatically** and attempts to **self-replicate** and **spread copies of itself over a network**. ● Medium of transmission ➢ Does not attach itself to a file ➢ Infects a vulnerable network ➢ Email attachments masquerading as a legitimate file ● Worms eat system resources of the host network by **consuming bandwidth and overloading web servers** to make the system **slow in speed** to the extent it stops responding.

4.3.2 Network Protection Mechanisms

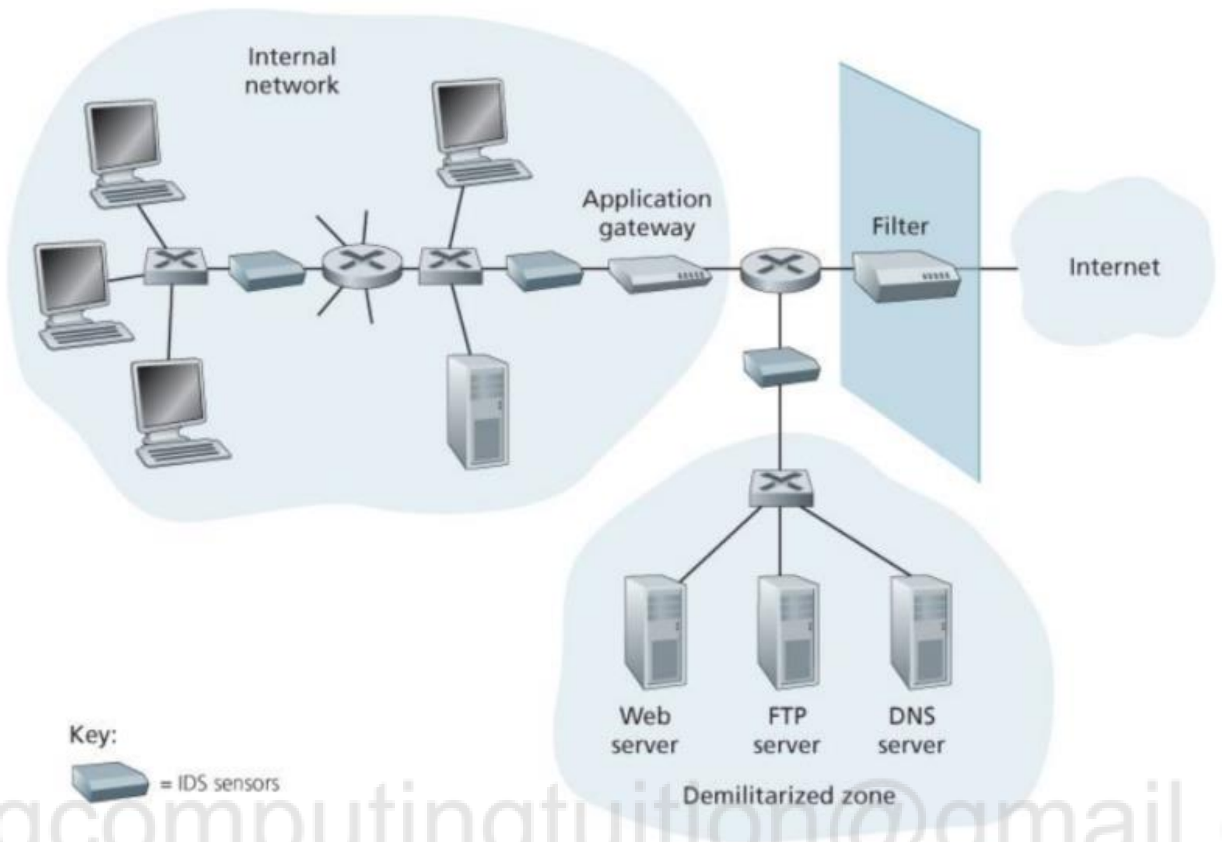
Overview of protection mechanisms categorised by each property of secure communication			
Confidentiality (Authorization/Access control methods)	**Message integrity**	**End-point authentication**	**Operational Security**
Only the sender and intended receiver should be able to understand the contents of the transmitted message. Is your permission granted?	The content of their communication is not altered, either maliciously or by accident, in transit.	Only authorised personnel can access the network or the data kept in the server. **Verify** the **identity (who?)** of a person in a situation when personal data needs to be accessed.	Networks owned by organisations are protected against cyberattacks.
• Encryption • File Permission	• Cryptographic hash functions • Digital signature	• Password • Security tokens • Biometrics	• Firewalls • IDS • IPS

Encryption	**File Permission**
Definition: Process of encoding information **End-to-end encryption** • Messages sent to one another are encrypted such that a third-party in between is not able to decrypt or read the contents of communication. • Only the authorised users have the secret key indicated by "https" protocol and padlock icon. **Types** • Symmetric key cryptography • Public key cryptography In public key cryptography, a pair of keys is used: public key and private key. Public key is known to anyone in the world while private key is only known to each specific user for their own use. **How does Alice send encrypted messages to Bob using public key cryptography?** 1. Alice uses Bob's public encryption key K_B^+ to encrypt her message into a ciphertext, $K_B^+(m)$ and sends it to Bob. 2. Upon receiving the encrypted message from Alice, Bob uses his private encryption key K_B^- to decrypt it by computing $K_B^-(K_B^+(m)) = m$	Definition: **Settings to control the ability of users** to view or edit specific files or folders. **Roles of administrators** • Administrators have the special ability to override the permissions for any file or folder • They also give permissions to individual users regarding access to certain files and folders. **Pitfalls** • Access to file/folder can be unintentionally granted to unauthorised users due to human error. • Impossible to stop intruders with physical access to storage devices.

Cryptographic hash functions	Digital signature
Definition: A function on bit string, m which computes a **fixed-size string H(m)** known as a hash. H(m) satisfies the following properties: - (One-way) It is computationally infeasible to find any input that maps to any new pre-specified output. - (Collision resistant) It is computationally infeasible to find any two distinct inputs that map to the same output. - Thus, it is computationally infeasible for an intruder to forge the contents of another message that has the same hash value as the original message. **How is shared secret (authentication key) used with hash functions?** **1.** Alice creates message m, concatenates a shared secret, **s** with **m** to create **m+s**, and calculates the hash **H(m+s)** known as the message authentication code (MAC) **2.** Alice then appends the MAC to m, creating an extended message **(m, H(m+S))** and sends it to Bob. **3.** Bob receives the extended message (m.h) and knowing s, calculates the MAC, H(m+s). If H(m+s)=h, message integrity is secured.	Definition: A digital signature is a **cryptographic technique to indicate the owner or creator** of a resource or to signify one's agreement with a document's content in a digital world. It **ensures the integrity of data and authenticates the sender**. Typically, we use a hash function to create a digital signature. **How is digital signature used together with cryptographic hash functions?** 1. Hash algorithm takes a message of arbitrary length and computes a fixed-length hash, H(m). 2. The sender uses his **private key, K_S^-** to encrypt the hash to the digital signature, that is, $K_S^-(H(M))$. 3. Both the message and the digital signature are sent to the receiver. 4. The receiver uses the **sender's public key** to decrypt the digital signature back to the sender's version of the hash. 5. The receiver uses the same hash algorithm to create a new hash from the received message, m. 6. If the two hashes match, it means the data is not altered and is sent by the known sender. Thus, message integrity is secured.

Password	Security Tokens	Biometrics
n-factor authentication: n represents the number of authentication methods used E.g. 2 Factor Authentication (2FA): User needs to provide password followed by a OTP		
How to have strong and secret **passwords**? - Hard-to-guess passwords that are a mixture of lower-case and upper-case letters, numbers and special characters. - Avoid reusing passwords but instead, use unique passwords for each account - Avoid leaving passwords unchanged for a long time but instead, update them at least every 90 days.	One Time Password (OTP) is generated from the **security token** or a mobile phone that the user owns. OTP is usually a string of seemingly random numeric or alphanumeric characters which is valid only for a short period of time.	**Biometrics** is based on the measurement of human characteristics which are unique to the individual and difficult to be replicated. E.g. fingerprint, voice, iris, face ID

Firewalls	Intrusion Detection System (IDS)	Intrusion Prevention System (IPS)
A firewall is a combination of hardware and software that **isolates an organisation's internal network from the Internet** at large, allowing some packets to pass and blocking others. **Function** 1. Firewall uses a set of pre-configured rules to examine traffic before it can be authorised and granted access to the network. 2. Firewall protects the LAN by performing **packet filtering** which is to allow or block the traffic based on the packet's source address, the destination address and the application protocols. **Limitations** ● Firewall cannot protect against internal sabotages. ● Firewall policies can be very restrictive and can limit users from performing legitimate operations.	A network security system that generates ***alerts*** when it ***observes suspicious / potentially malicious traffic***. **Function** 1. IDS **compares each header of a passing packet with several signatures**. Each signature is a set of rules pertaining to an intrusion activity such as the packet's source and destination port numbers. 2. To detect many attack types, IDS systems also ***perform deep packet inspection***, that is, to look beyond the header fields and into actual ***application data*** in the packet. 3. When multiple IDS sensors are deployed, they **typically work in concert**, sending information about suspicious traffic activity to a **central IDS processor**, which collects and integrates the information and sends alarms in time to network administrators to take further protective actions. **Limitations** ● IDS can generate false alarms. ● IDS requires someone to respond to the generated alerts.	A network security system that ***filters out*** suspicious and potentially malicious traffic. **Limitations** ● IPS can be easily overloaded by network traffic. ● IPS has low accuracy when network traffic is encrypted.
	Network can be partitioned into 2 regions: a high-security region further protected by an application gateway and a lower-security region also known as a demilitarised zone (DMZ),	

Other Protection mechanisms (from O-levels)

Install anti-virus & anti-spyware	● Frequent scanning of antivirus programs detects malwares, stops them from running and removes them.
Update software regularly	● Malware usually exploits bugs and loopholes in software such as web browsers, operating systems and software on network devices. ● Keeping these softwares updated regularly will patch up newly discovered bugs and loopholes before any malware can use these to gain unauthorised access to a computer.
Configure browser settings	● Enable spam filters on the email service to block emails from suspicious sources. ● Change web browser settings to allow cookies only from certain trusted websites or simply disable all cookies. ● Clear web browser history and cookies after very browsing sessions.