Our Computing Notes

1.1: Algorithmic Representation	Practice/Context	Tick here ✅
1.1.1 Pseudocode		✅
1.1.2 Flow Chart		✅
1.1.3 Control Structures		✅
1.1.4 Decision Tables		✅
1.1.5 Modular Design		✅
1.2: Fundamental Algorithms	**Practice/Context**	**Tick here ✅**
1.2.1 Sorting Algorithms		✅
1.2.3 Search Algorithms		✅
1.2.5 Time Complexity		✅
1.3: Data Structures	**Practice/Context**	**Tick here ✅**
1.3.1 Memory Allocation		✅
1.3.3 Stack, Linear Queue, Circular Queue		✅
1.3.5 Linked List		✅
1.3.6 Search Trees		✅
1.3.7 Traversal		✅
2.2: Programming Elements	**Practice/Context**	**Tick here ✅**
2.2.5 Recursion		✅
2.2.6 Code Tracing		✅
2.2.7 Call Stack		✅
2.4: Program Testing	**Practice/Context**	**Tick here ✅**
2.4.1 Data Validation, Data Verification		✅
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)		✅
2.5.3 Unified Modelling Language (UML) Diagram		✅
3.1-3.2: Data Representation	**Practice/Context**	**Tick here ✅**
3.1.1 Number Base Conversion		✅
3.2.1 Character Encoding		✅
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 Client-Server Architecture, Peer-to-Peer Network		✅
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 Security Defences		✅
4.3.3 Encryption, Digital Signature and Authentication		✅

1.1 Algorithmic Representation

1.1.1 Pseudocode

Cheatsheet

Comments	//
1D Array of Integers	ARRAY [1: 5] OF INTEGERS
2D Array of Integers	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

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
	ENDFOR

def FooOne(n):
  return(n)
	FUNCTION FooOne(n) RETURNS INTEGER
	  RETURN n
	ENDFUNCTION

def FooTwo(n):
  print(n)
	PROCEDURE FooTwo(n)
	  OUTPUT n
	ENDPROCEDURE

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

Commented [1]: *Inclusive* of "5".

Commented [2]: less than or more than

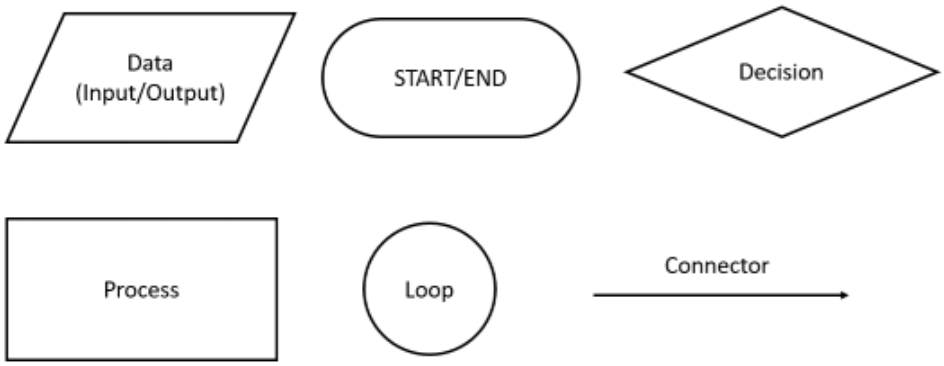
Commented [3]: Since keyword used is *TO*, it is *inclusive* of n.

Commented [4]: *FUNCTION*: have return
PROCEDURE: no return

Commented [5]: *FUNCTION*: have return
PROCEDURE: no return

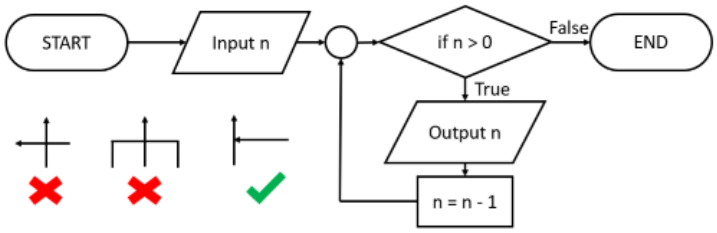
1.1.2 Flow Chart

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

Commented [6]: *Flow Chart* illustrates clearly the 3 fundamental programming constructs: *Sequence*, *Selection* and *Iteration*.

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

1.1.4 Decision Tables

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

Example

Conditions:

- English >= B4
- Chinese >= B4
- GPA with 3 other subjects > 2.0

Outcomes:

- Approved
- Conditionally approved
- Rejected

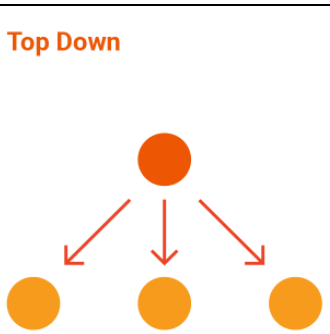
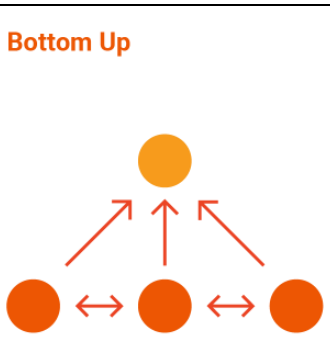
CONDITIONS	**C1**	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**	**C8**
English >= B4	Y	Y	Y	Y	N	N	N	N
Chinese >= B4	Y	Y	N	N	Y	Y	N	N
GPA with 3 other subjects > 2.0	Y	N	Y	N	Y	N	Y	N
OUTCOMES								
Approved	X							
Conditionally approved			X		X			

Reduced Decision Table

CONDITIONS	**C1**	**C2/4**	**C3**	**C5**	**C6**	**C7/8**
English >= B4	Y	Y	Y	N	N	N
Chinese >= B4	Y	-	N	Y	Y	N
GPA with 3 other subjects > 2.0	Y	N	Y	Y	N	-
OUTCOMES						
Approved	X					
Conditionally approved			X	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		

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 of sub-problems. When similar sub-problems occur, previous solutions can be reused.

Modular programming is the process of breaking down a program into separate sub-programs.

Advantages	Disadvantages
Can be stored in a library to be reused in other solutions.	High time complexity.
Easier to debug, maintain and modify.	High space complexity.

	Top-down testing	Bottom-up testing
Definition	Higher level modules are tested first, followed by lower level modules.	Lower level modules are tested first, followed by higher level modules.
Observation	More difficult to observe test results.	Easier to observe test results.
Identification of Errors	More difficult to identify errors in the critical modules.	Easier to identify errors in the critical modules.
Complexity	Simple and less data intensive.	Complex and more data intensive.

1.2 Fundamental Algorithms

1.2.1 Sorting Algorithms

Bubble Sort

- Simple sort
- Repeated swapping of adjacent elements if they are in the wrong order.
- Time complexity:
  - Worst case: $O(n^2)$
  - Average case: $O(n^2)$
  - Best case: $O(n)$

```
def bubble_sort(lst):
    for _ in range(len(lst) - 1):
        for i in range(len(lst) - 1):
            if lst[i] > lst[i + 1]: # right < left
                lst[i], lst[i + 1] = lst[i + 1], lst[i]
    return lst
```

Insertion Sort

- Simple sort
- Repeated insertion of elements to their correct positions one at a time.
- Time complexity
  - Worst case: $O(n^2)$
  - Average case: $O(n^2)$
  - Best case: $O(n)$

```
def insertion_sort(lst):
    for i in range(1, len(lst)):
        temp = lst[i]
        j = i - 1
        while temp < lst[j] and j >= 0: # when temp < left, move left
            lst[j + 1] = lst[j]
            j -= 1
        lst[j + 1] = temp
    return lst
```

Quick Sort

- Efficient sort
- Pick a pivot, partition the other elements into 2 sub-arrays that are higher or lower than the pivot, and sort recursively.
- Time complexity
  - Worst case: $O(n^2)$
  - Average case: O(nlogn)
  - Best case: O(nlogn)

Ideal pivot: Median of array → partitions into 2 sub-arrays of equal size with each pass

- For arrays that are sorted, nearly sorted, or reverse sorted, choosing the first or last item as the pivot would lead to worst case $O(n^2)$ time complexity.
- Choose pivot at random → ↓probability of choosing worst case pivot → ↓time complexity

```python
def quick_sort(lst):
    if len(lst) <= 1: # base case
        return lst
    else: # general procedure
        pivot = lst[0]
        lower, higher = [], []
        for item in lst[1:]:
            if item < pivot:
                lower.append(item)
            else:
                higher.append(item)
        return quick_sort(lower) + [pivot] + quick_sort(higher)
```

Merge Sort

- Efficient sort
- Divide array into two sub-arrays recursively, then merge all sub-arrays recursively.
- Time complexity
    - Worst case: O(nlogn)
    - Average case: O(nlogn)
    - Best case: O(nlogn)

```python
def merge_sort(lst):
    def merge(left, right): # merge left and right
        result = []
        while left and right:
            if left[0] < right[0]:
                result.append(left.pop(0))
            else:
                result.append(right.pop(0))
        return result + left + right

    if len(lst) <= 1: # base case
        return lst
    else: # general procedure
        mid = len(lst) // 2
        left = merge_sort(lst[:mid])
        right = merge_sort(lst[mid:])
        return merge(left, right)
```

Comparisons

- For sorted or nearly sorted arrays:
    - **Bubble sort:** If smallest item is the last item or largest item is the first item → each O(n) pass only swaps positions → need to swap n times → ↑comparisons → worst case $O(n^2)$ time complexity
    - **Insertion sort:** Position of item will only be changed if it is in the wrong position → only few O(n) passes needed to insert the items that were in the wrong position → ↓comparisons → best case O(n) time complexity
    - **Quick sort:** If pivot is smallest or largest item → array is divided into 2 sub-arrays of unequal sizes → need to divide array n times → partitioning into 2 sub-arrays requires O(n) time complexity → ↑comparisons → worst case $O(n^2)$ time complexity

- **Merge sort:** O(nlogn) time complexity for all cases

1.2.3 Search Algorithms

Unordered Linear Search

- To search an array that is not sorted.
- Time complexity: O(n)

```
def unordered_linear_search(item, lst):
    for i in range(len(lst)):
        if lst[i] == item:
            return True
    return False
```

Ordered Linear Search

- To search an array that is sorted.
- Time complexity: O(n)

```
def ordered_linear_search(item, lst):
    for i in range(len(lst)):
        if lst[i] == item:
            return True
        elif lst[i] > item:
            return False
    return None
```

Binary Search

- To search an array that is sorted.
- Time complexity: O(logn)

```
# Iterative Binary Search
def binary_search(lst, item):
     low = 0
     high = len(lst) - 1

     while low <= high:
          mid = (low + high) // 2
          if lst[mid] == item:
               return mid
          elif item < lst[mid]:
               high = mid - 1
          else:
               low = mid + 1
     return -1
```

```python
# Recursive Binary Search
def binary_search(lst, item):
    def helper(lst, item, low, high):
        midpoint = (high + low) // 2
        if low > high:
            return False
        elif lst[midpoint] == item:
            return True
        elif item > lst[midpoint]:
            return helper(lst, item, midpoint + 1, high)
        else:
            return helper(lst, item, low, midpoint - 1)
    return helper(lst, item, 0, len(lst) - 1)
```

Hash Table Search

Hashing is a technique used to store and retrieve data as quickly as possible.

- Time complexity
  - Worst case: O(n)
  - Average case: O(1)
  - Best case: O(1)

Hash table is a data structure that stores the keys to their associated values.

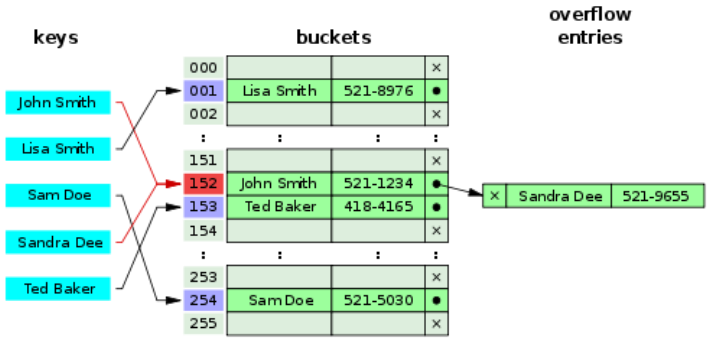
Hash function is the function used to transform the key into its hash value.

Collision is the condition where two keys have the same hash value.

Collision resolution is the process of finding an alternative location when collision occurs.

Collision resolution techniques

- **Direct chaining (open hashing):** Keys with the same hash value are stored in a linked list.

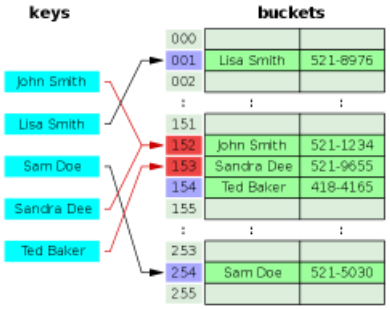


- **Open addressing (closed hashing):** Keys with the same hash value are stored in alternative locations in the hash table.
  - **Linear probe:** Intervals between probes are set to 1.

Commented [7]: Note:
Hash table is a slot-key pair, unlike dictionaries, which has a key-value pair.

Commented [8]: For *string* keys, use the *sum of all ASCII characters*.

    - `h+1, h+2, h+3...`
  - **Quadratic probe:** Intervals between probes increase quadratically.
    - $h+1^2$, $h+2^2$, $h+3^2$...
  - **Double hashing:** 2 hash functions are used to compute the hash value.
    - `h1(key) + h2(key), h1(key) + 2 * h2(key), h1(key) + 3 * h2(key)...`



Direct Chaining	Open Addressing
High space complexity.	Keys have to be unique.

Good hash function

- ↓collisions
- ↓clustering
- Time efficient
- Uses all information provided in the key.

$$load\ factor = \frac{self.count}{self.size}$$

- ↑load factor → ↑collisions + ↑clustering → ↑time complexity
- **Solution:** Expand the hash table and rehash all keys.

Hash Table	
Slot	Key
1	
2	
3	
4	
5	

Commented [9]: Time complexity: O(1)

Commented [10]: Generally when load factor > 0.75

Commented [11]: Only do this if there are more data to be inserted into the hash table.

6	

1.2.5 Time Complexity

Big O notation is a rough measure of resources used by a computational process.

Commented [12]: Coefficients and constants are ignored.

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

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.
Implementation	Stack memory is used.	Heap memory is used.
Flexibility	Static memory allocation is inflexible in allocating memory.	Dynamic memory allocation is flexible in allocating memory.
Efficiency	Static memory allocation is time efficient.	Dynamic memory allocation is time inefficient.
Memory	Static memory allocation does not allow memory to be reused.	Dynamic memory allocation allows memory to be reused.
Example	• **Array:** Elements are stored in a single continuous block of memory.	• **Linked list:** Nodes are stored in different parts of the memory and are linked up using pointers.

Commented [13]: Reduce *memory wastage*.

1.3.3 Stack

- Last In First Out (LIFO)
- **Operations:** Push, pop, peek

Applications of stack

- **Reverse Polish Notation (RPN)**
  - RPN is used to represent arithmetic expressions in postfix form and store them in a stack for evaluation later.

infix	*postfix*	*prefix*
(a + b) * c	a b + c *	* + a b c
a + (b * c)	a b c * +	+ a * b c

Infix form : <identifier> <operator> <identifier>
Postfix form : <identifier> <identifier> <operator>
Prefix form : <operator> <identifier> <identifier>

- **Call stack**
  - Call stack is used to keep track of recursive calls.
  - Stack frame is used to store local variables, parameters provided, and the return address.
  - When a function is called, a stack frame is created and is pushed to the top of the stack.
  - When a function returns a value, its stack frame becomes invalid and is popped from the top of the stack.
  - Function is called recursively until there are no more stack frames in the call stack.

Implementation

```
class Stack:
      def __init__(self):
            self.stack = []

      def push(self, data):
            self.stack.append(data)

      def pop(self):
            if self.stack:
                  return self.stack.pop()

      def peek(self):
            if self.stack:
                  return self.stack[-1]

      def size(self):
            return len(self.stack)
```

1.3.3 Queue

- First In First Out (FIFO)
- **Operations:** Enqueue, dequeue, peek

Applications of queue

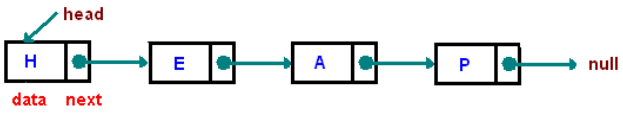
- **Router queue**
  - Router can only process one packet at a time. If packets arrive faster than the router can process them, these packets would be enqueued at the end of the router queue.
  - Packets are dequeued from the start of the router queue so that they will continue to be transmitted in the original order.

	Linear Queue	Circular Queue
Definition	Data is arranged linearly.	Data is arranged circularly. Front and rear ends are connected to each other.
Flexibility	Enqueue and dequeue operations can only be done at the rear and front ends respectively.	Enqueue and dequeue operations can be done at any position.
Efficiency	Linear queue is time inefficient.	Circular queue is time efficient.
Memory	Linear queue is space inefficient.	Circular queue is space efficient.

Implementation

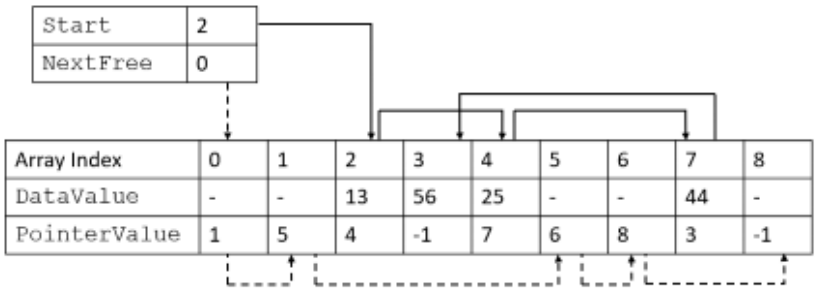
```
class Queue:
      def __init__(self):
            self.queue = []

      def enqueue(self, data):
            self.queue.append(data)

      def dequeue(self):
            if self.queue:
                  return self.queue.pop(0)

      def peek(self):
            if self.queue:
                  return self.queue[0]

      def size(self):
            return len(self.queue)
```

1.3.5 Linked List



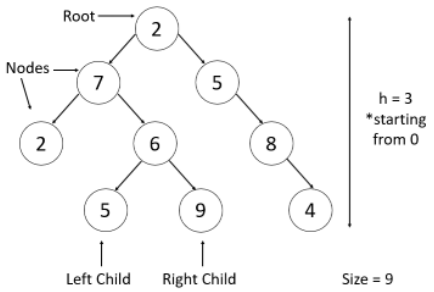
Advantages	Disadvantages
Flexible in allocating memory.	High time complexity for traversal.
Memory can be reused.	High space complexity.

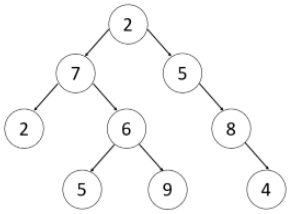
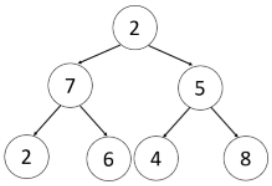
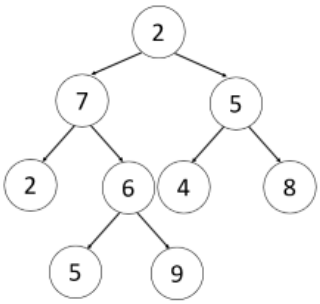
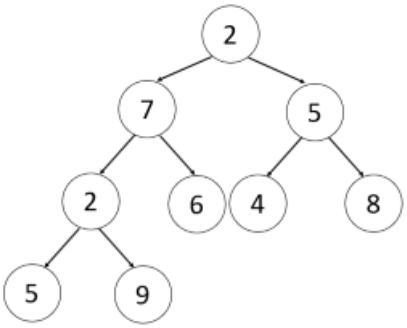
Implementation using free space list



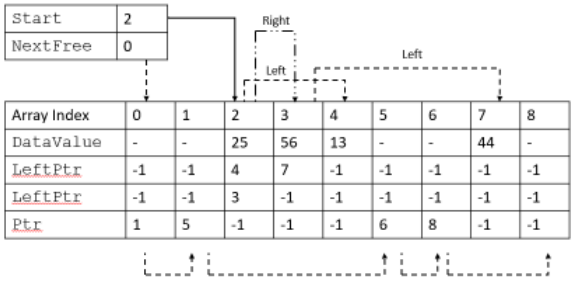
1.3.6 Search Trees

Binary Tree

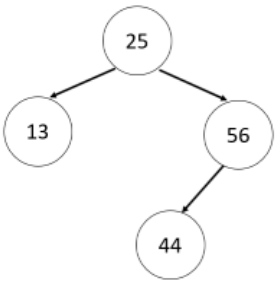


Rooted Binary Tree A binary tree in which it has a root node and every node has at most two children. 	**Perfect Binary Tree** A binary tree in which all interior nodes have two children and all leaves have the same depth 
Full Binary Tree A binary tree in which every node in the tree has either 0 or 2 children. 	**Complete Binary Tree** A binary tree in which every level, except possibly the last, is completely filled, and all nodes in the last level are as far left as possible. 

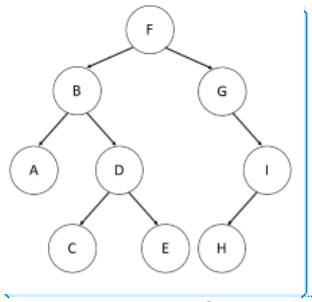
Implementation using free space list



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



this is the tree 🌳 btw

Commented [14]: height = 3

Inorder()

- Used to obtain value of all nodes in ascending order.

[Left, Display, Right]

1. Traverse the left subtree recursively
2. Display current node if search reaches end of the leaf
3. Traverse the right subtree recursively

Order: A,B,C,D,E,F,G,H,I

Time Complexity: O(n)

Preorder()

- Used to create a copy of the BST.

[Display, Left, Right]

1. Display the root node (or current node from a subtree)
2. Traverse the left subtree recursively
3. When search reaches the end of the leaf, search will traverse the right subtree recursively

Order: F,B,A,D,C,E,G,I,H Time Complexity: O(n)
Postorder() ● Used to delete a tree from the leaf node to the root node. ***[Left, Right, Display]*** 1. Traverse the left subtree recursively 2. Traverse the right node if right node is available 3. Display the current node if search reaches the end of searched path Order: A,C,E,D,B,H,I,G,F Time Complexity: O(n)

Applications of BST

- **Huffman encoding**
  - Compress text by converting each character into their associated Huffman code.
  - Characters that are more commonly used are found closer to the root node.
  - Add 0 to Huffman code if node is on left, and add 1 to Huffman code if node is on right.

	Hash Table	Binary Search Tree
Sorting	No operation that allows us to arrange all keys in ascending order.	Inorder traversal allows us to arrange all keys in ascending order.
Searching	Hash table does 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.
Efficiency	Hash table has a time complexity of O(1) for average case, but O(n) for worst case.	BST has a time complexity of O(logn) for all cases.

2.2 Programming Elements

2.2.5 Recursion

Recursion is a method of program design that repeatedly breaks down a problem into smaller subproblems until one or more terminating cases are reached.

Example

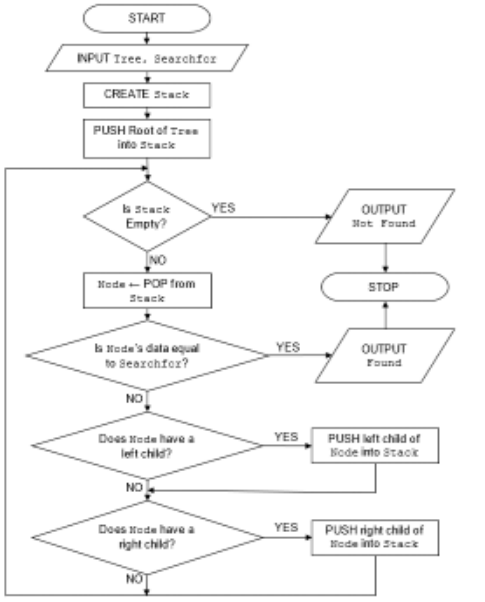
```
def summation(lst, count = 0):
    if len(lst) == 0: # base case
        return count
    else: # general procedure
        return summation(lst[1:], count + lst[0])
```

2.2.6 Code Tracing

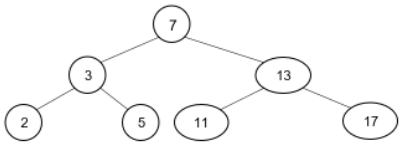
Trace tables are visual representations of code tracing.

Case	Variable 1	Variable 2	Variable 3
1	True	1	1
2	True	0	1
3	False	0	0

Commented [15]: *Note*: Recursive calls are not output. Output refers to print statements.



(i) Given the input `Tree` below and a `Searchfor` value of 5, draw a trace table to illustrate the algorithm. [5]



Stack	Node	Output
[7]	7	
[3, 13]	13	
[3, 11, 17]	17	
[3, 11]	11	
[3]	3	
[2, 5]	5	Found

The contents of `b_tree` is shown below. −1 represents the null pointer.

Index	l_ptr	data_item	r_ptr
0	−1	A	−1
1	0	+	2
2	−1	B	−1
3	1	*	5
4	−1	C	−1
5	4	−	6
6	−1	D	−1

root
3

A procedure, `P`, uses recursion.

```
01 PROCEDURE P(Index: INTEGER)
02   IF b_tree[Index].l_ptr <> -1 THEN
03      P(b_tree[Index].l_ptr)
04   ENDIF
05   IF b_tree[Index].r_ptr <> -1 THEN
06      P(b_tree[Index].r_ptr)
07   ENDIF
08   OUTPUT b_tree[Index].data_item
09 ENDPROCEDURE
```

(d) Copy and then complete the trace table for procedure `P` showing the values of `Index` and the output.

`Index`	Output
1	

[5]

Index	Output
1	
0	A
2	B
	+

(Note to students: recursive calls are not output. In procedure P, only line 8 is the output)

2.2.7 Call Stack

Call stack

- Call stack is used to keep track of recursive calls.
- Stack frame is used to store local variables, parameters provided, and the return address.
- When a function is called, a stack frame is created and is pushed to the top of the stack.
- When a function returns a value, its stack frame becomes invalid and is popped from the top of the stack.
- Function is called recursively until there are no more stack frames in the call stack.

2.4 Program Testing

2.4.1 Data Validation and Data Verification

Data Validation is the process of ensuring that data is clean, correct, and useful.

Type of Data Validation	Function	Example
Presence	Checks for input	`len(user_input) != 0`
Type	Checks if input is of correct data type	`user_input.isalpha() == True` `user_input.isdigit() == True`
Range	Checks if input is in desired range	`int(user_input) > 0`
Length	Checks if input is of correct length	`len(user_input) == 8`
Layout/Format	Checks if input is in a correct format (such as dates or identification number)	`user_input[0] == "T" or user_input[0] == "S"`
Restricted Value	Checks if input is in predefined set of expected values	`user_input in (0, 4, 8, 16)`
Check Digit / Check Code	E.g. NRIC, Vehicle Registration Number	Weighted Modulus

Reasons for using check digit:

- **Transcription error** occurs when a character is typed incorrectly.
- **Transposition error** occurs when position of characters are reversed.

Modulus 11 Check Digit

Step 1	Multiply each digit of the original number by its weight at that position, and find the sum of these products. **Example:** 02757

Position	**0**	**1**	**2**	**3**	**4**
Weights	6	5	4	3	2

$$(0 \times 6) + (2 \times 5) + (7 \times 4) + (5 \times 3) + (7 \times 2) = 67$$

Step 2	Take the result from step 1 and modulo 11. **Example:** $67 \ (mod \ 11) = 1$
Step 3	Subtract the result from step 2 from 11 . **Example:** $11 - 1 = 10$
Step 4	If result in step 3 is 10, it is replaced by X. Otherwise, keep the result as it is. This

Commented [16]: Examples include *International Standard Book Number (ISBN) 10*, where the final character of a ten-digit ISBN is a check digit computed so that multiplying each digit by its position in the number (counting from the right) and taking the sum of these products modulo 11 is 0.

	result would be the original number's check digit, and is added to the far right of the original number. **Example:** $02757X$

Reasons for using string over integer for check digits:

- When the check digit = 10, it has 2 digits and needs to be replaced by X.
- Leading zeros are allowed in check digits.

Data verification is the process of ensuring that the copy of data is equal to its original.

Examples

- Proofreading
- Double entry

2.4.3 Errors and Debugging

Errors

Syntax error occurs when the code violates grammar rules of the programming language.

```
>>> 5>>>4
SyntaxError: invalid syntax
```

Logic error occurs when the program runs successfully but returns an undesired result.

➔ Requires manual tracing

```
>>> from math import pi
>>> def area_of_circle(radius):
     area = pi * (radius ** 3) # wrong formula for area of circle
     return area

>>> print(area_of_circle(4))
201.06192982974676
```

Runtime error occurs when the program can be compiled, but cannot be executed as intended.

➔ Performing operations on incompatible types

➔ Accessing non-existent objects

```
>>> 100/0
Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    100/0
ZeroDivisionError: division by zero
```

```
>>> "1" + 0
Traceback (most recent call last):
```

Commented [17]: To check whether the 6 digits are valid:

Check digit has weight of 1.
Replace X with 10.
Multiply each digit of the original number by its weight at that position, and find the sum of these products.
If result is divisible by 11, then 6 digits are valid.

Example:
(0*6)+(2*5)+(7*4)+(5*3)+(7*2)+(10*1)=77

Commented [18]: *Example:* int("0111") = 111

```
  File "<pyshell#1>", line 1, in <module>
    "1" + 0
TypeError: can only concatenate str (not "int") to str
```

```
>>> a + b
Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    a + b
NameError: name 'a' is not defined
```

```
>>> lst = [1, 2, 3]
>>> lst[3]
Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    lst[3]
IndexError: list index out of range
```

Debugging techniques
• Print statements • Try-except blocks • Breakpoints in shell/IDEs

2.4.4 Program Testing

	Black Box Testing	White Box Testing
Definition	Checks whether the output of the program follows the function of the program without knowledge of software internals. It is carried out from the perspective of a user.	Uses knowledge of software internals to test out every possible path. It is carried out from the perspective of the developer.
Efficiency	Black box testing is time efficient for large amounts of code.	White box testing is time inefficient for large amounts of code.
Code Access	Black box testing does not require code access.	White box testing requires code access.
Coverage	Black box testing provides limited coverage.	White box testing provides maximum coverage.

User Acceptance Testing (UAT) checks whether the product satisfies the user's needs.

Alpha testing is the stage of testing that is done by testers in the organisation.

Beta testing is the stage of testing that is done by a selected number of real users.

Type of Data	Definition
Normal Data	Data within accepted range.
Boundary Data	Data which are at absolute limits of accepted range.
Invalid Data	Data outside accepted range.

2.5 Object Oriented Programming (OOP)

2.5.1 Object Oriented Programming (OOP)

Class is a blueprint that defines the properties and behaviours of an object.

Instance is an object created from a class. It is a usable entity that carries actual values of its properties.

Attributes are descriptions.	**Accessors** are getters.
Methods are functions.	**Mutators** are setters.

`self` references the current instance of the object.

`__init__()` initialises the object's states.

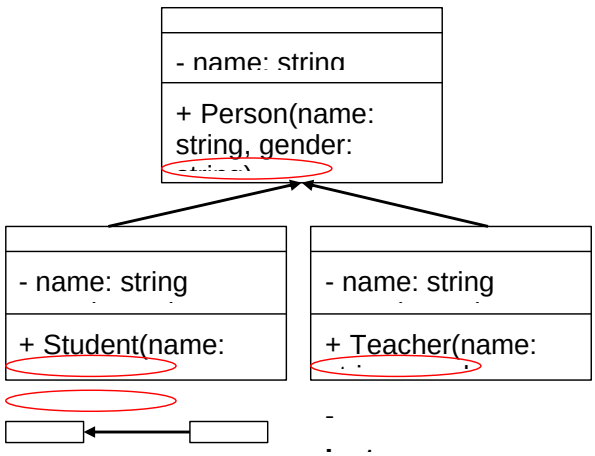
Encapsulation is the bundling of data with methods that operate on that data. Internal representation is hidden from the public.

+ Allows behaviour of objects to be hidden.
+ Protect internal state of objects from being corrupted.

Inheritance is where subclass retains similar implementations of attributes and methods from superclass. This promotes software reuse.

Polymorphism is where the function of methods in subclass can be changed while keeping the same name. This promotes code generalisation.

2.5.3 Unified Modelling Language (UML) Diagram



3.1-3.2 Data Representation

3.1.1 Number Base Conversion

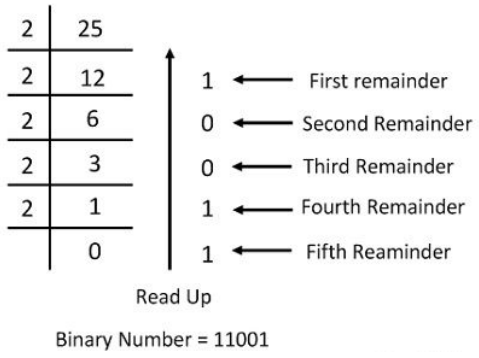
Conversion from base N to base 10

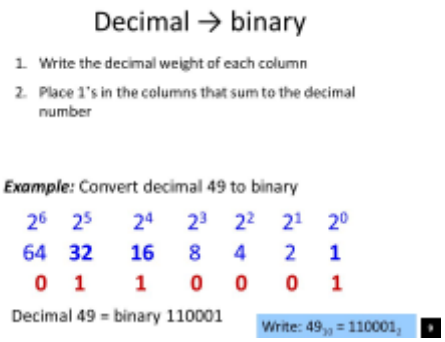
Binary	1	1	1	0	1	0	1
Denary	1×2^5	1×2^4	1×2^3	0×2^2	1×2^1	0×2^0	1×2^{-1}

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

Conversion from base 10 to base N

Repeated Division	Sum of Weights





When converting from binary to base-4 or base-8 or base-16, group up the binary numbers into groups of 2 for base-4, groups of 3 for base-8, or groups of 4 for base-16.

Base-2	1	1	1	0	1	0	0
→	$1 \times 2^{0+6}$	$1 \times 2^{2+3}$	$1 \times 2^{1+3}$	$1 \times 2^{0+3}$	1×2^2	0×2^1	0×2^0
→	$(2^0) \times 8^2$	$(2^2 + 2^1) \times 8^1$			$(2^2) \times 8^0$		
Base-8	1	6			4		

$$1110100_2 = 164_8$$

Base-2	1	1	1	0	1	0	0
→	$1 \times 2^{2+4}$	$1 \times 2^{1+4}$	$1 \times 2^{0+4}$	1×2^3	1×2^2	0×2^1	0×2^0
→	$(2^2 + 2^1 + 2^0) \times 16^1$			$(2^2) \times 16^0$			
Base-16	7			4			

$$1110100_2 = 74_{16}$$

3.2.1 Character Encoding

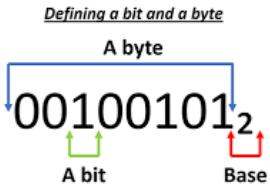
American Standard Code for Information Interchange (ASCII)

- Character encoding standard for electronic communication. 7-bit character code where each value represents a unique character.
- **Numbers:** 48-57
- **Upper case:** 65-90
- **Lower case:** 97-122

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

Unicode

- Computing industry standard for consistent encoding, representation, and handling of text.
  - **UTF-8:** 8-bit character code whose first 128 characters are the same as ASCII.



Bit (b): Stores 0 or 1.
Byte (B): 8 bits.

Commented [19]: Can also be interpreted as *logical values* (true/false), *algebraic signs* (+/-), *activation states* (on/off).

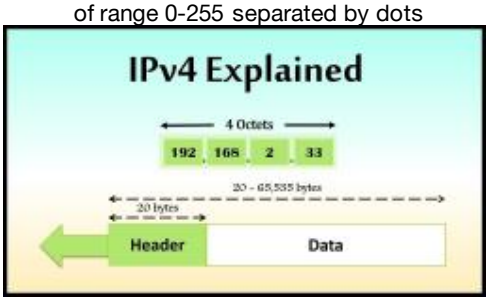
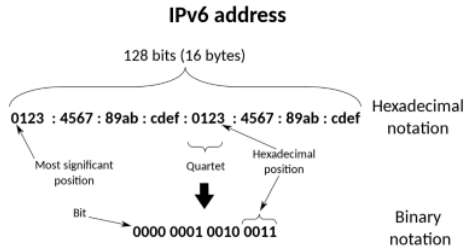
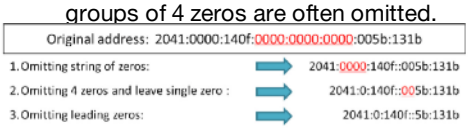
RGB Code is a model where primary colours are added together to form a wide range of colours.

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

Commented [20]: *MAC address* handles the *physical connection* from computer to computer.

IP address handles the *logical routable connection* from both computer to computer AND network to network.

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  • 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  • For compactness, leading zeros and groups of 4 zeros are often omitted. 

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

Commented [21]: AB:CD:EF:12:34:56

- 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

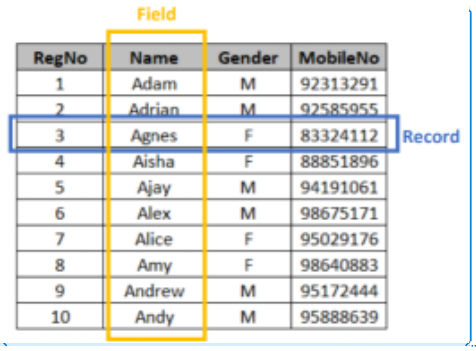
3.3 SQL Database

Commented [22]: *SQL*: Structured Query Language

3.3.1 Relational Database Management System (RDBMS)

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

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



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

Commented [23]: *Tables* are made up of records and fields.

Record: row
Field: column

3.3.2 Keys

Commented [24]: *Note:* When giving the definition of primary, secondary and composite keys, the definition of a candidate key must be given first.

A **candidate key** is an attribute or a combination of attributes that can uniquely identify each record.

A **primary key** is a candidate key that is most suited to become the main key.

A **secondary key** is a candidate key that is not chosen as the primary key.

A **composite key** is a combination of two or more attributes that can uniquely identify each record.

Commented [25]: *Composite key* is a subset of *candidate key*.

A **foreign key** is an attribute in one table that references the primary key in another table.

3.3.3 Data Redundancy, Data Inconsistency, Data Dependency

Reasons to use RDBMS over flat files such as .txt and .csv:

- **Data redundancy** is where the same data is being stored more than once.
- **Data inconsistency** is where different versions of the same data exist at the same time.

Data dependency:

1. **Functional dependency** is a direct relationship where one attribute uniquely identifies another attribute in the same table.
2. **Transitive dependency** is an indirect relationship formed by two functional dependencies.

Commented [26]: X -> Z is a transitive dependency if the following three functional dependencies hold true:

* X->Y
* Y does not ->X
* Y->Z

3.3.4 Normalisation

Normalisation is the process of organising the tables in a database to reduce data redundancy and inconsistency.

1. **Unnormalized Form (UNF)**

2. **First Normal Form (1NF)**
   Conditions:
   - All attributes should hold only atomic values (each record has to be unique).

3. **Second Normal Form (2NF)**
   Conditions:
   - In 1NF
   - All non-key attributes should be fully dependent on the entire primary key.

4. **Third Normal Form (3NF)**
   Conditions:
   - In 2NF
   - No transitive dependencies.

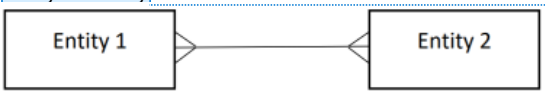
3.3.5 Entity-Relationship (ER) Diagram

Commented [27]: *Useful tip:* "Chicken feet" is where the foreign key is at, i.e. Primary Key -< Foreign Key

1. One to One

2. One to Many

3. Many to Many



Commented [28]: Many to many relationships should always be decomposed into a one-many-one relationship.

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

Legend:
___: Primary Key
*: Foreign Key

Commented [29]: If the question uses ___ as the foreign key, and an attribute is both the primary and foreign key, *solid underline should be written over the dotted underline*.

1. UNF
   StudentCCAInfo (MatricNo, Name, Gender, CivicsClass, CivicsTutor, HomeRoom, CCAName1, CCATeacherIC1, CCAName2, CCATeacherIC2, CCAName3, CCATeacherIC3)

2. 1NF

StudentCCAInfo (MatricNo, Name, Gender, CivicsClass, CivicsTutor, HomeRoom, CCAName, CCATeacherIC)

3. 2NF
   Student (MatricNo, Name, Gender, CivicsClass, CivicsTutor, HomeRoom)
   StudentCCA (**MatricNo***, **CCAName***)
   CCAInfo (CCAName, CCATeacherIC)

4. 3NF
   Student (MatricNo, Name, Gender, **CivicsClass***)
   Civics (CivicsClass, CivicsTutor, HomeRoom)
   StudentCCA (**MatricNo***, **CCAName***)
   CCAInfo (CCAName, CCATeacherIC)

3.3.8 SQL Query

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

1. **CREATE TABLE**

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

CREATE TABLE 'Employee' (
'EmployeeID' INTEGER UNIQUE NOT NULL 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')
)
```

   a. **PRIMARY/COMPOSITE KEY**

```
PRIMARY KEY('PersonID', 'EmployeeID')
```

b. FOREIGN KEY

```
FOREIGN KEY('PersonID') REFERENCES Person('PersonID')
```

c. CHECK

```
CHECK(Age > 0 AND Age < 100)
```

2. INSERT INTO

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

3. SELECT FROM

```
SELECT *
FROM Person
```

a. DISTINCT

```
SELECT DISTINCT Person.Name
FROM Person
```

b. IS NOT NULL

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

c. LIKE

```
WHERE Person.Name LIKE 'J%'
OR Person.Name LIKE '%s'
```

d. ORDER BY

```
ORDER BY Person.Age ASC
```

e. GROUP BY

```
GROUP BY Person.Name
```

f. LIMIT

```
LIMIT 5
```

g. 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
```

Commented [30]: For Table *Company*

Commented [31]: For Table *Person*

Commented [32]: If Table has 5 names [Tom, Tom, Jesus, Bob, Jerry]:
This query would only return [Tom, Jesus, Bob, Jerry]

Commented [33]: Select names that start with J or end with s

Commented [34]: ASC/DESC

Commented [35]: Query returns only maximum of 5 rows.

Commented [36]: Remember to use *GROUP BY*!

6. **DROP TABLE**

```
DROP TABLE Company
```

3.3 NoSQL Database

3.3.6 SQL vs NoSQL

	SQL Database	NoSQL Database
Definition	Relational Database Management System (RDBMS).	Non-relational Database Management System.
Schema	SQL databases have a predefined schema that prevents changes to be made easily.	NoSQL databases have a dynamic schema that allows changes to be made easily.
Scalability	SQL databases are vertically scalable, making it more expensive to increase performance.	NoSQL databases are horizontally scalable, making it cheaper to increase performance.
Data Integrity	SQL databases promote data integrity.	NoSQL databases do not promote data integrity.
Query	SQL databases support complex queries.	NoSQL databases do not support complex queries.

3.3.7 Applications of SQL and NoSQL

SQL Database	NoSQL Database
Data integrity is required.	New features are added frequently.
Complex queries are required.	Large amounts of data are stored.
Ensures **ACID** compliance. **A:** Atomicity **C:** Consistency **I:** Isolation **D:** Durability	

3.3.8 NoSQL Query

1. **use database**

```
> use database
```

2. **createCollection**

```
> db.createCollection("crimes")
```

3. **show dbs**

```
> show dbs
admin     0.000GB
database  0.000GB
local     0.000GB
```

4. **show collections**

```
> show collections
crimes
```

5. **insert**

```
> db.crimes.insert({"_id": 11, "accused": "Pablo
Skittles", "gender": "M", "age": 33, "records": [{"type":
"Kidnapping", "victims": ["Beluga"]}]})
```

6. **find**

```
> db.crimes.find()
```

   a. **$gt/lt/eq/gte/lte**

```
> db.crimes.find({"age": {"$gt": 30}}, {"accused": 1,
"_id": 0})
```

   b. **$and, $in/nin, .$.**

```
> db.crimes.find({"$and": [{"records.type": {"$in":
["Kidnapping", "Terrorism"]}}, {"gender": "M"}]},
{"records.$.victims": 1, "_id": 0})
```

   c. **$and, $in, .$.**

```
> db.crimes.find({"$or": [{"records.victims": {"$size":
2}}, {"nationality": {"$exists": False}}]}, {"accused": 1,
"_id": 0})
```

7. **count**

```
> db.crimes.count()
```

   a. **find → count**

```
> db.crimes.find({"records.type": "Murder"}).count()
```

8. **sort**

```
> db.crimes.find({}, {"accused": 1, "age": 1, "_id":
0}).sort({"age": 1, "accused": -1})
```

9. **updateMany ($set/$unset)**

```
> db.crimes.updateMany({}, {"$set": {"nationality":
"Singapore"}})
```

10. **update ($set/$unset)**

```
> db.crimes.update({"accused": "Pablo Skittles"},
{"$unset": {"nationality": 0}})
```

11. **deleteMany**

```
> db.crimes.deleteMany({"nationality": "Singapore"})
```

12. **delete**

Commented [37]: Names of accused with age > 30

Commented [38]: Names of victims of kidnapping or terrorism incidents caused by male accused(s)

Commented [39]: Names of accused who have crime record that involves 2 victims or do not have nationality

Commented [40]: Number of accused that have committed murder.

Commented [41]: Note that this is different from pymongo implementation.

Commented [42]: Sort by age of accused in ascending order, followed by name of accused in descending order.

Commented [43]: Note that this is different from pymongo implementation.

```
> db.crimes.delete({"accused": "Pablo Skittles"})
```

13. **drop**

```
> db.crimes.drop()
```

14. **dropDatabase**

```
> db.dropDatabase()
```

Commented [44]: Drops the collection.

3.3 Data Protection

3.3.9 Data Privacy, Data Integrity

Data privacy is the requirement for data to be accessed by authorised people only.

Data integrity is the requirement for data to be accurate and up-to-date.

3.3.10 Methods to Protect Data

Data Privacy	Data Integrity
Encrypt data.	Backup data.
Use strong passwords.	Validate input data.

3.3.11 Backup, Archive

Backup is a copy of data used to protect against data loss.

Archive is a copy of data made for long-term storage and reference.

Backups **restore** and archives **retrieve**.

3.3.12 Version Control, Naming Convention

Version control is a system that allows software teams to keep track of changes over time.

- Allow for rollbacks when mistakes are made.
- Promotes team collaboration.
- **Example:** Git

Naming convention is a set of rules for choosing the character sequence for identifiers.

- Allow people to understand code easily.

3.3.13 Personal Data Protection Act (PDPA)

Personal data refers to data that can uniquely identify each person.

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

	Personal Data Protection Act (PDPA)
Consent	Organisations can only collect, use and disclose personal data after consent is given by the individual.
Accountability	Organisations must make information about their personal data protection policies available.
Notification	Organisations must notify the individual of the purpose for collecting, using or disclosing their personal data.

3.4 Ethics

3.4.1 Code of Conduct

	Code of Conduct
Professionalism	• Use computing skills to benefit society. • Do not harm society for personal benefit.
Integrity	• Be truthful about one's competence. • Be impartial when giving advice.
Responsibility	• Complete assigned work before the due date. • Acknowledge mistakes and correct them when they are made.
Competence	• Continue to improve computing skills. • Increase public knowledge about computing.

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 → ↑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

Commented [45]: *Structural unemployment (UnE):* UnE due to mismatch of skills between job seekers and job openings.

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.

	Intranet	Virtual Private Network (VPN)
Definition	Private network within an organisation that usually excludes outsider access.	Encrypted connection over the Internet from a device to a network.
Purpose	Provides members of the organisation with resources that are unavailable to the public.	Ensures sensitive data is securely transmitted to a network.

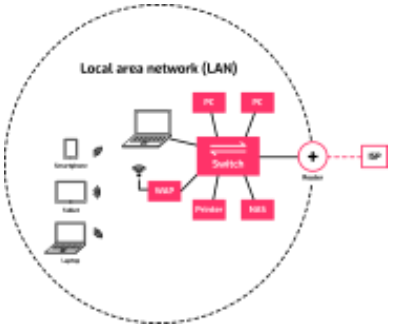
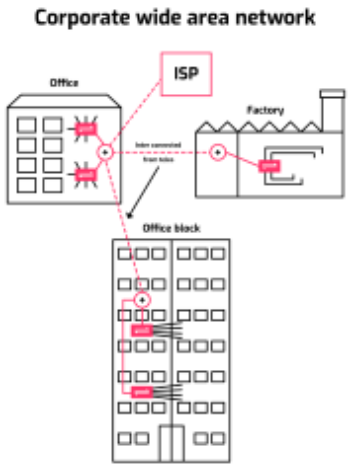
The Internet is a worldwide network of networks that uses the Internet Protocol (IP) suite.

World Wide Web (WWW) is a network of public web pages available on the Internet.

	Local Area Network (LAN)	Wide Area Network (WAN)
Definition	Computer network that connects computers over a small geographical area. Examples include the WiFi and Ethernet.	Computer network that connects computers over a large geographical area. Examples include 4G and the Internet.
Basic Networking Devices	• Hub • Switch • Bridge	• Router

Commented [46]: Examples are needed in the definition for *Local Area Network (LAN)* and *Wide Area Network (WAN)*.

Commented [47]: Small geographical area: house, school, office.

Visualisation		

OSI Model	TCP/IP Model	Basic Networking Devices
Application Layer	Application Layer	
Presentation Layer		
Session Layer		
Transport Layer	Transport Layer	
Network Layer	Internet Layer	**Router** transmits packets between Local Area Networks (LANs). • **Router** maintains a **route table** so that it can transmit packets to the correct interface based on the destination address in the Internet Protocol (IP) packet header.
Data Link Layer	Network Access Layer	**Switch**/**Bridge** transmits frames to the intended device. • **Switch** maintains a **switch table** so that it can transmit frames to the correct port based on the MAC address.
Physical Layer		**Hub** broadcasts incoming bits from one port to all of its other ports.

Commented [48]: OSI Model mnemonic:
Please (Physical Layer)
Do (Data Link Layer)
Not (Network Layer)
Throw (Transport Layer)
Sausage (Session Layer)
Pizza (Presentation Layer)
Away (Application Layer)

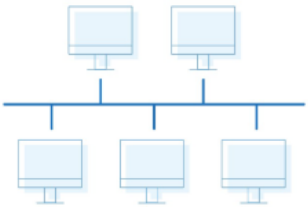
Commented [49]: *OSI Model*: Open Systems Interconnection Model

Commented [50]: Do note in the OSI Model that Physical Layer is Layer 1, and Application Layer is Layer 7!

Commented [51]: TCP/IP Model Acronym: *NITA*
N: Network Access Layer
I: Internet Layer
T: Transport Layer
A: Application Layer

Commented [52]: *Switch* is basically a multiport *bridge*, and this makes the switch faster and more efficient.

Commented [53]: *Switch table* is also known as a *MAC address table*.

<table>
<tr><th colspan="3">Network Topology</th></tr>
<tr><td>Point-to-Point (P2P)</td><td>Two devices are directly connected.</td><td>  </td></tr>
<tr><td>Bus Topology</td><td>All devices are connected to a main cable via drop lines.
<table><tr><td>Cost</td><td>Bus topology is cheap.</td></tr><tr><td>Installation</td><td>Bus topology is easy to install.</td></tr><tr><td>Efficiency</td><td>Bus topology is time inefficient.</td></tr><tr><td>Reliability</td><td>Bus topology is unreliable as the failure of the bus leads to failure of the network.</td></tr></table></td><td>  </td></tr>
<tr><td>Ring Topology</td><td>Each device is connected to two other devices.
<table><tr><td>Cost</td><td>Ring topology is cheap.</td></tr><tr><td>Installation</td><td>Ring topology is easy to install.</td></tr><tr><td>Efficiency</td><td>Ring topology is time inefficient.</td></tr><tr><td>Reliability</td><td>Ring topology is unreliable as the failure of one device leads to failure of the network.</td></tr></table></td><td>  </td></tr>
</table>

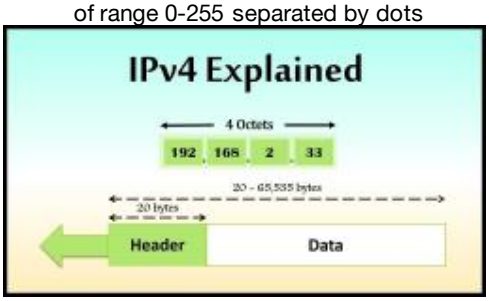
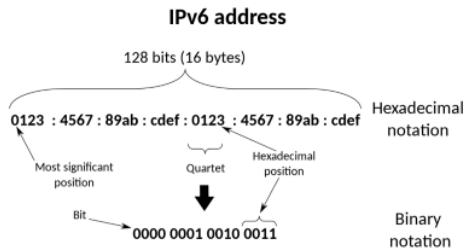
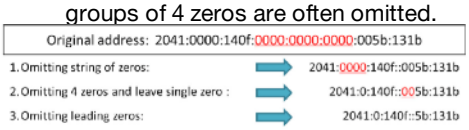
<table>
<tr>
<td>Star Topology</td>
<td>All devices are connected to a central networking device.
<table>
<tr><td>Cost</td><td>Star topology is expensive.</td></tr>
<tr><td>Installation</td><td>Star topology is easy to install.</td></tr>
<tr><td>Efficiency</td><td>Star topology is time efficient.</td></tr>
<tr><td>Reliability</td><td>Star topology is unreliable as the failure of the central networking device leads to failure of the network.</td></tr>
</table>
</td>
<td> 
</td>
</tr>
</table>

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.

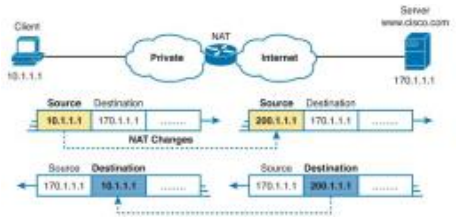
Two kinds of IP addresses are used today:

Commented [54]: IPv4 addresses are more frequently encountered than IPv6 addresses.

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  • 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  • For compactness, leading zeros and groups of 4 zeros are often omitted. 

Solutions to the IPv4 address exhaustion problem:

- IPv6
- **Network Address Translation (NAT)** is the method of mapping one IP address space into another while the packet is in transit across a router.



Uniform Resource Locator (URL) is a reference to a web resource that specifies its location on a computer network.

Protocol	Third-level domain	Second-level domain	Top-level domain	Directory name	File name

http://	www	.example	.com	/directory	/file.txt

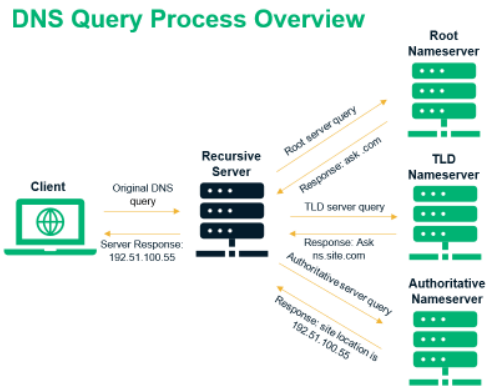
Domain Name System (DNS) server is a decentralised and hierarchical database that converts URLs to their associated IP addresses.

Types of DNS Services

	Authoritative DNS	Recursive DNS
Definition	Service that answers DNS queries by converting domain names into IP addresses.	Service that takes URL requests from users and checks the records obtained from authoritative DNS servers for the associated IP address.
Example	Amazon Route 53	Google Public DNS

DNS query process

1. Client opens the web browser and types the URL in the address bar.
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 to obtain information on the top-level domain (TLD).
4. Recursive server sends the query again to a TLD name server to obtain information on the second-level domain (SLD).
5. Recursive server sends the query again to an authoritative name server to obtain information on the domain and its associated IP address.
6. Authoritative name server sends the IP address back to the recursive server.
7. Recursive server sends the IP address back to the client through the web browser. Recursive server also caches the IP address for the domain.



Commented [55]: *Recursive DNS* are also known as *DNS resolvers*. This is usually managed by the users' *Internet Service Provider (ISP)*.

Commented [56]: This is so that the recursive server can respond more quickly the next time someone browses a webpage with the same domain.

Commented [57]: This cached data is only stored for a predetermined period of time, and is not indefinite.

4.1.3 Communication Protocols

Communication protocols are a set of rules that define the format, order of information exchanged, and the actions taken after information is transmitted over the network.

- Allow networking devices to communicate with each other.
- Allow communication to be initiated and terminated.

Reasons for use of communication protocols:

- **Hardware side:** Communication protocols allow different networking devices made by different manufacturers to communicate with each other.
- **Software side:** Communication protocols allow different applications to run simultaneously without affecting one another.

OSI Model	TCP/IP Model	Protocol Data Unit
Application Layer	Application Layer	Data
Presentation Layer		
Session Layer		
Transport Layer	Transport Layer	Segments
Network Layer	Internet Layer	Packets
Data Link Layer	Network Access Layer	Frames
Physical Layer		Bits

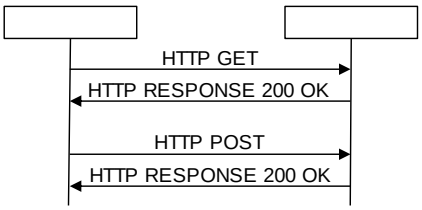
TCP/IP Model	
Application Layer	Responsible for providing access to network resources.
Transport Layer	Responsible for process-to-process data transfer.
Internet Layer	Responsible for transmitting packets between computer networks.
Network Access Layer	Responsible for transmitting frames and bits between devices connected to a computer network.

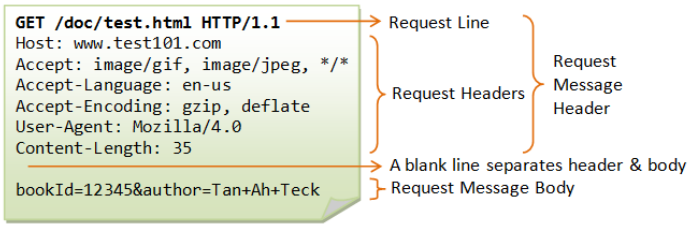
TCP/IP Model	
Application Layer	• **Hypertext Transfer Protocol (HTTP)** is a protocol that allows hypertext to be exchanged between nodes.

Commented [58]: with higher level protocols like HTTP, FTP, SMTP

Commented [59]: with TCP(reliable) and UDP(unreliable)

Commented [60]: For *every item embedded in the html webpage*, such as image, css or javascript, an *additional HTTP request* is needed.





- File Transfer Protocol (FTP)
- Simple Mail Transfer Protocol (SMTP)
- Domain Name System (DNS)

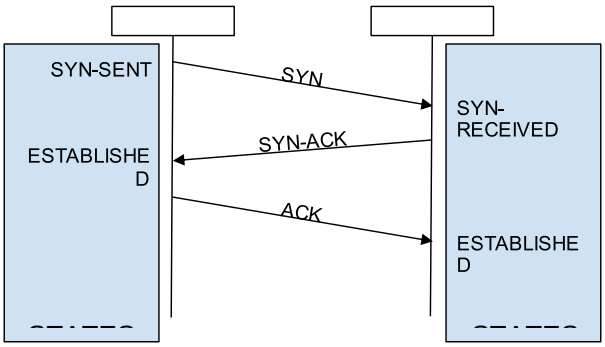
Transport Layer

	Transmission Control Protocol (TCP)	User Datagram Protocol (UDP)
Definition	TCP is a connection oriented protocol that requires a connection to be established before transmitting segments.	UDP is a connectionless protocol that transmits datagrams continuously and disregards any issues on the receiving end.
Efficiency	TCP is time inefficient.	UDP is time efficient.
Reliability	TCP is reliable as it supports error checking.	UDP is unreliable as it does not support error checking.
Order	TCP segments are transmitted in a specific order.	UDP datagrams are not transmitted in a specific order.
Retransmission	TCP segments can be retransmitted.	UDP datagrams cannot be retransmitted.
Application	• Email • File upload	• Gaming • Streaming

TCP 3-way handshake:

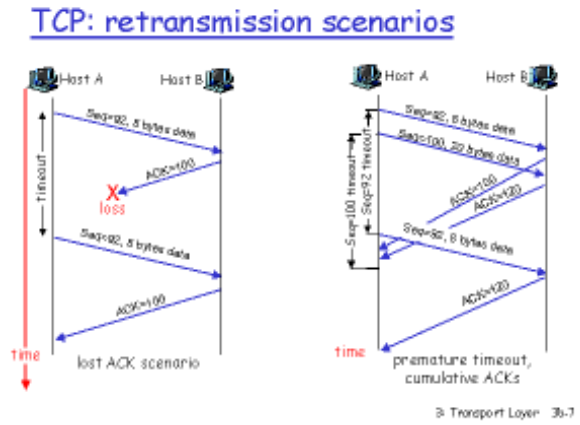
Commented [61]: For UDP, order of data is managed by the *Application Layer*.

Commented [62]: 3-way handshake shows how TCP connection is established.

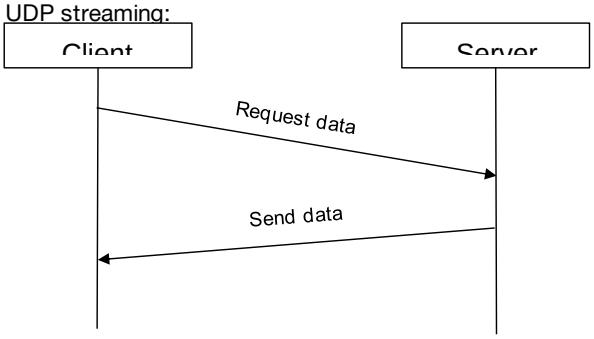


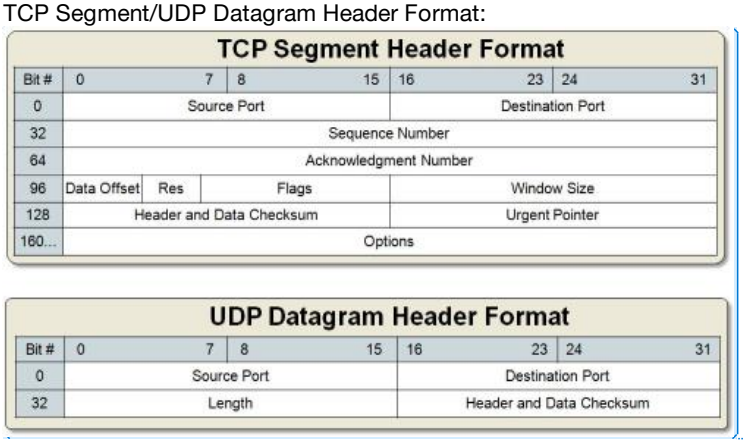
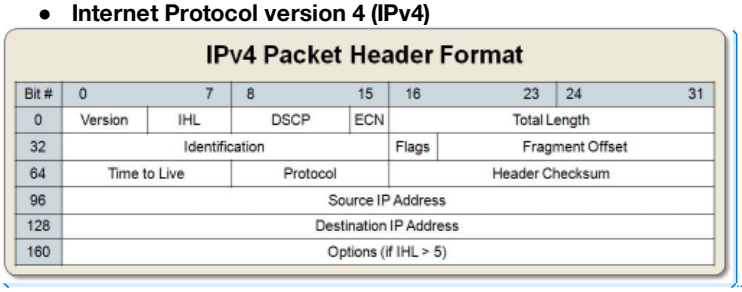
Synchronize (SYN) is sent to server to request for connection to be established.
Acknowledge (ACK) is sent to server to acknowledge response of server.

TCP retransmission:



UDP streaming:



	TCP Segment/UDP Datagram Header Format:  **TCP Segment Header Format** Bit # 0 7 8 15 16 23 24 31 0 Source Port | Destination Port 32 Sequence Number 64 Acknowledgment Number 96 Data Offset | Res | Flags | Window Size 128 Header and Data Checksum | Urgent Pointer 160... Options **UDP Datagram Header Format** Bit # 0 7 8 15 16 23 24 31 0 Source Port | Destination Port 32 Length | Header and Data Checksum
Internet Layer	• **Internet Protocol version 4 (IPv4)**  **IPv4 Packet Header Format** Bit # 0 7 8 15 16 23 24 31 0 Version | IHL | DSCP | ECN | Total Length 32 Identification | Flags | Fragment Offset 64 Time to Live | Protocol | Header Checksum 96 Source IP Address 128 Destination IP Address 160 Options (if IHL > 5) • Internet Protocol version 6 (IPv6) • Internet Control Message Protocol (ICMP)
Network Access Layer	• **Media Access Control (MAC)** transmits frames via the Network Interface Card (NIC) or remotely shared channels. • Ethernet

Commented [63]: As you can see, UDP Datagram only does a *checksum* for error checking, making it unreliable.

Commented [64]: *IHL*: Internet Header Length
DSCP: Differentiated Services Code Point
ECN: Explicit Congestion Notification

Commented [65]: *Media Access Control (MAC)* is also known as *Medium Access Control*.

Commented [66]: MAC is a *sublayer* of the Data Link Layer in the OSI model.

4.1.4 Circuit Switching, Packet Switching

	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.
Delay	Circuit switching has call setup delay.	Packet switching has packet transmission delay.

Commented [67]: *Connection-oriented:* TCP
Connectionless: UDP, IP

Commented [68]: *UDP:* Packet transmission delay
TCP: Call setup delay + packet transmission delay

Methods to reduce transmission delay:

- **Increase transmission capacity**
- **Compress data**
- **Quality of Service (QoS):** Feature of routers where priority can be given to packets that are more important.

Commented [69]: For packet switching only.

4.1.5 Client-Server Architecture, Peer-to-Peer Network

	Client-Server Architecture	Peer-to-Peer (P2P) Network
Definition	Computing model where server provides the service and client requests for the service. Server has higher privileges than the client.	Computing model where all peers provide and request for the service. All peers have equal privileges.
Management	Client-Server Architecture is easy to maintain.	P2P network is difficult to maintain.
Security	Client-Server Architecture is secure.	P2P network is insecure.
Cost	Client-Server Architecture is expensive to maintain.	P2P network is cheap to maintain.
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.

4.2 Web Application

4.2.1 Web Application, Native Application

	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.
Efficiency	Web applications are time inefficient.	Native applications are time efficient.
Internet Access	Web applications require Internet access.	Native applications do not require Internet Access.
Cost	Web applications are cheap to maintain.	Native applications are expensive to maintain.
Accessibility	Web applications can be accessed by many operating systems.	Native applications can only be accessed by a few operating systems.

4.2.2 Usability Principles

Usability Principles

1. Consistent
2. Simple
3. Responsive
4. Easy navigation
5. Feedback about progress

4.3 Network Security

4.3.1 Information Security Threats

Malware is a malicious software that intentionally causes damage to a computer or network.

CIA Triad	
Confidentiality	Ensures information is accessible only by authorised individuals.
Integrity	Ensures information is reliable.
Availability	Ensures information is available when it is accessed.

Malware	Confidentiality (C)	Integrity (I)	Availability (A)
Virus is a type of malware that replicates itself by inserting its own code into other resources when executed by a human. Example: • Creeper • Melissa	✅	✅	✅
Worm is a type of malware that replicates itself by inserting its own code into other resources without human activation. Example: • ILOVEYOU • WannaCry	✅	✅	✅
Trojan horse is a type of malware that aims to take control of the computer by disguising itself as legitimate software. Example: • Shedun • MEMZ	✅	✅	✅

Cyber-attack	Confidentiality (C)	Integrity (I)	Availability (A)
Denial of Service (DoS) is a type of cyber-attack that aims to shut down the network by flooding it with traffic.			✅

Commented [70]: *CIA Triad* is a security model that highlights core data security objectives and serves as a guide for organizations to keep their sensitive data protected from unauthorized access and data exfiltration.

Commented [71]: *Viruses* cannot spread without human action, but *worms* can.

Commented [72]: *WannaCry* is a cryptoworm that asks for ransom in exchange for decrypting the hosts' files.

Example: • Mirai Botnet DDoS attack, 2016			
Phishing is a type of cyber-attack that aims to steal user data by social engineering the user. Example: • Nigerian Prince Email Scam	☑		
Internet Protocol (IP) Spoofing is 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.	☑		
Packet Sniffing is a type of cyber-attack that aims to steal data by capturing data packets that are transmitted through a network.	☑		
Bots is a type of software that performs automated tasks on command. Malicious bots allow an attacker to remotely take control over the computer. Example: • Mirai Botnet DDoS attack, 2016	☑		☑

Code Injection Attack	Confidentiality (C)	Integrity (I)	Availability (A)
SQL Injection is a type of code injection attack that aims to modify, expose, and destroy existing data, by injecting SQL code to interfere with the queries sent to a SQL database. Example: • UNION attack	☑	☑	☑
Cross Site Scripting (XSS) is a type of code injection attack that aims to steal resources by injecting Javascript code into the webpage.	☑		☑

4.3.2 Network Security Defences

AAA Framework	
Authentication	Verification of a user's identity.
Authorisation	Assignment of privileges to a user.
Auditing	Tracking of a user's activity.

Commented [73]: *Internet Protocol (IP) Spoofing* is more of a method to carry out cyber-attacks by bypassing security.

Commented [74]: *Authentication, Authorisation and Auditing (AAA) Framework* refers to a common security framework for mediating network and application access.

Commented [75]: *Authentication, Authorisation and Auditing* are also separate examples of network defences.

Security controls are protections to reduce security risks to resources.

Security Controls	Examples
Physical Security Control	Fences, doors, locks, security cameras, fire extinguishers.
Administrative Security Control	Incident response processes, security awareness, training.
Technical Security Control	User authentication, antivirus software, firewalls.
Legal Security Control	Privacy laws, policies, clauses.

Network Security Defences	
Firewall	**Firewall** is a network security system that only allows authorised network traffic to pass based on the security rules of the organisation. Types of Firewalls • **Packet Filtering Firewall** filters Internet Protocol (IP) packets based on the source and destination address in the IP packet header. • **Stateful Inspection Firewall** filters Internet Protocol (IP) packets based on the state and context of the network connection. Limitations • Firewall cannot protect against internal sabotages. • Firewall policies can be very restrictive and can limit users from performing legitimate operations.
Intrusion Detection System (IDS)	**Intrusion Detection System (IDS)** is a network security system that detects suspicious activities and generates alerts when they are detected. Types of IDS • **Host-based Intrusion Detection System (HIDS)** is an IDS that is deployed on a host. • **Network-based Intrusion Detection System (NIDS)** is an IDS that is deployed across a network. • **Signature-based Intrusion Detection System (SIDS)** is an IDS that requires signatures to detect intrusion. • **Anomaly-based Intrusion Detection System (AIDS)** is an IDS that requires anomalies to detect intrusion. Limitations

Commented [76]: *Proxies:* Server that acts as an intermediary between servers and clients

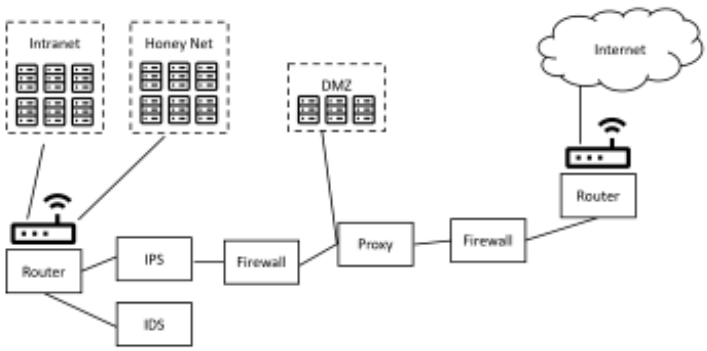
Gateway: Proxy server that passes unmodified requests and responses, it is sometimes a tunneling proxy

Forward proxy: Internet-facing (open) proxy used to retrieve data from a wide range of sources

DMZ: physical or logical subnetwork that contains and exposes an organization's external-facing services to an untrusted network, usually a larger network like the Internet.
External networks can only access what is exposed in the DMZ while the rest of the organization's network is firewalled.
If design is effective, allows organization extra time to detect and address breaches before they would further penetrate into the internal networks

Honeypots: Network of computers constructed for the purpose of luring hackers.
The honeypot diverts them away from authentic resources and network administrators deploy intrusion detection systems and other monitoring systems to discover the identity of the hackers and some of the techniques they are using.

Commented [77]: Example: Member of organisation cooperates with external cyber-attacker.

	• IDS can generate false alarms. • IDS requires someone to respond to the generated alerts.
Intrusion Prevention System (IPS)	**Intrusion Prevention System (IPS)** is a network security system that detects suspicious activities and prevents them when they are detected. Types of IPS • **Host-based Intrusion Prevention System (HIPS)** is an IPS that is deployed on a host. • **Network-based Intrusion Prevention System (NIPS)** is an IPS that is deployed across a network. Limitations • IPS can be easily overloaded by network traffic. • IPS has low accuracy when network traffic is encrypted.
Visualisation	

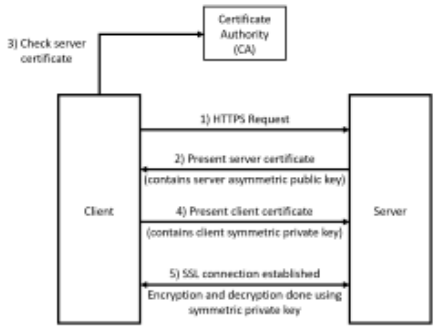
	Firewall	Intrusion Detection System (IDS)	Intrusion Prevention System (IPS)
Basis of Filtering/Detection	Organisation security rules	Cyberthreat database	Cyberthreat database
Protection	Active protection	Generate alerts	Active protection

Commented [78]: *IDS/IPS* compare network packets to a *cyberthreat database* containing known signatures of cyberattacks and flag any matching packets.

Data at rest refers to data that is stored physically.
Data in transit refers to data that can be actively transmitted between devices.

Data Protection At Rest	Data Protection In Transit
• Advanced Encryption Standard (AES) is a symmetric cryptographic algorithm. • Rivest-Shamir-Adleman (RSA) is an asymmetric cryptographic algorithm.	• **Secure Sockets Layer (SSL)/Transport Layer Security (TLS)** is a protocol for transmitting information securely over the Internet.

Commented [79]: *SSL* is now deprecated, in favour of *TLS*.

	• **Hypertext Transfer Protocol Secure (HTTPS)** is a HTTP protocol that is encrypted using SSL/TLS. HTTPS process: 

4.3.3 Encryption, Digital Signature and Authentication

Other Network Security Methods	

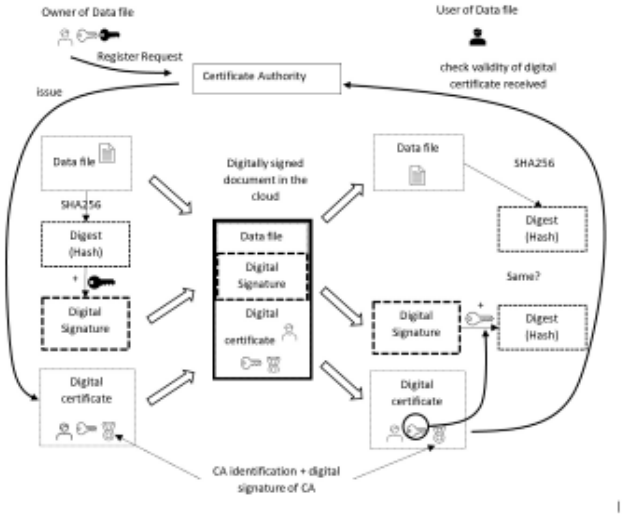
Encryption	**Encryption** is the process of encoding information.
	
	Types of Encryption Methods
	• Data At Rest
	  ○ Symmetric Cryptographic Algorithm
	  ■ Advanced Encryption Standard (AES)
	  ■ Data Encryption Standard (DES)
	  ■ Fernet
	  ○ Asymmetric Cryptographic Algorithm
	  ■ Rivest-Shamir-Adleman (RSA)
	• Data In Transit
	  ○ Secure Sockets Layer (SSL)
	  ○ Transport Layer Security (TLS)
	  ○ Hypertext Transfer Protocol Secure (HTTPS)
	• Email
	  ○ Pretty Good Privacy (PGP)
	  ○ Secure/Multipurpose Internet Mail Extensions (S/MIME)

Digital Signature	**Digital Signature** is a process that guarantees that the contents of a message has not been altered in transit. How it Works 1. The sender computes a message digest (with an algorithm such as RSA or SHA1) and then encrypts the digest with their private key, which forms the digital signature

Commented [80]: Same public and private keys.

Commented [81]: Different public and private keys.

Commented [82]: *Digital Signature* employs asymmetric cryptographic algorithms.

	2. The sender transmits the digital signature with the message 3. The receiver decrypts the digital signature with the public key of the sender, thus regenerating the message digest 4. The receiver computes a message digest from the message data that was received and verifies that the two digests are the same. If these digests match, the message is both intact and authentic.
Digital Certificate	**Digital Certificate** is a digital document that is used to prove the ownership of a public key.  Types of Digital Certificates • Secure Sockets Layer (SSL) Certificate
Authentication	**Authentication** is the verification of a user's identity. Types of Authentication Methods • Passwords • Two Factor Authentication (2FA) • Biometric Authentication

Commented [83]: *Authentication* can be done using *Digital Certificates*!

Commented [84]: *Two Factor Authentication (2FA)* often uses a *One Time Password (OTP)*.

Commented [85]: *Biometric Authentication* often uses iris and fingerprint for authentication.