NETWORK(NW): A group of devices that are connected together to communicate and share resources

+: Shared resources; Easy access; Collaboration

-: High cost; Security breach; Failure; Maintenance.

WAN: NW that spans large area, even across countries

LAN: NW that is self-contained, spans small area

INTERNET: Largest global WAN, public owned, contains a large number of Intranets

INTRANET: Private NW that uses protocol & services to share company info. Can be accessed from Internet.

+: Security, privacy; speed; cost

SWITCHING: Tech^{n} to transmit data over NW to devices

Packet Switching	**Circuit Switching**
Has no established route	Establishes route to send
Sent on individual routes	Packets on same route
Not easily intercepted	Easily intercepted
Need to be reordered	Remain in correct order
Maximises use of network	Ties up large areas of NW

MODELS: Explain data communication processing in soft/hardware; blueprint for developers to build NWs

OSI ~: theoretical | **TCP/IP ~:** implemented in internet

PROTOCOL: Rules for data transmission agreed by sender, receiver. | E.g: data formats, error detection.

LAYERING: Simplifies model, standardised interface

TCP/IP [PROTOCOL] {PRO. DATA UNIT} <ADDRESSING>

Each layer only communicates with the adjacent layers.

At each layer, data unit formats; Headers with info in sender & removed from data unit in receiver's process

5 Application – High level FN to end users [HTTP/SMTP] {messages} <N/A>

Other protocols: FTP, SSH (file transfer) ; SMTP, POP3, IMAP4 (email) ; HTTP/HTTPS (webpage)

4 Transport – FN to transmit messages btw 2 programs

[TCP/UDP] {segment/ datagram} <port number>

Transmission Control Protocol (TCP): uses 3 way handshake to est connection before transmission. Data is broken up into segments with sequential no. to reassemble at receiver. Ensures security & validity of data.

***3 way handshake*:** [1] SYN [2] SYN/ACK [3] ACK
The client sends a SYN request to the server, Server sends back a SYN/ACK message to acknowledge. The client sends another ACK message to establish connection.

Sequencing: Data transmitted in segments, TCP places a seq. no. on EACH segment sent, so receiver can reorder.

TCP > UDP. Connection est. first -> used when high reliability necessary & transmission time is less critical.

Universal Datagram Protocol (UDP): Connectionless

+: No waiting time | **-:** Stability & reliability may be missing

UDP used when performance **>** receiving all data

Port no.: No. representing a process running on NW

Socket: Combination of IP address + port no. Identifier for an application process on NW

What is the need for comm protocols in a NW?
Protocol is a set of pre-agreed rules to govern communication between devices. It is necessary as there are different devices and networks. Hence there is a need for rules to specify to these differences are to be resolved, for communication to be possible.

3 Internet – FN to determine route btw 2 devices [IP] {packets} <ip address>

Internet Protocol (IP): responsible for routing individual datagrams and addressing.

IP Address: logical address that identifies devices on NW.

Network ID: the ID common to all devices on a network

Host ID: the unique ID of particular device on a NW

Device: Router – Uses IP addresses to send data packets to designated host. Maintains a routing table.

2 Data Link – FN to transmit packets between 2 devices in same NW. [Ethernet/WiFi] {Frames} <MAC address>

MAC: Media Access Control (physical address of device; unique for each machine)

Describes how devices format data for transmission to other devices

Device: Modem - Converts data from analog to digital (btw two transmission media); Switch – sends data frames to designated host, maintains a MAC table.

1 Physical – FN to transmit individual bits through a transmission medium [10 Base T/ 802.11] {Bits} <NA>

CLIENT-SERVER NW: one or more devices act as a server, other devices request service from server.

~ SERVER: Centralized storage area for resources; Provides service to other devices in NW, and is only dedicated to such tasks; Controls access to resources on NW, Filters network traffic

~ CLIENT: Sends request to server and server responds accordingly; Does not share any resources.

~ +: Faster traffic; more reliable – if 1 client down others still can use; Access rights of client is controlled; Resources secured on server and can be updated/ backup easily

~ -: If server down, whole network is down. Centralized server more expensive to build, requires maintenance

---

PEER-TO-PEER NW: All devices interact and share resources with any other device without going through a server. They share the responsibility and capability.

MAIL SERVERS: Handle the transfer of messages to and from other mail servers and email clients.

SMTP: Send ; **POP3:** Receive & Store @ Client ; **IMAP:** Receive & Store @ Server, SYN on devices, Better org.

SMTP: SMTP server at sender's email service recognises the recipient's address' domain server, and connects to it with DNS. Message will then be handed to POP3

POP3: The protocol copies the emails and stores it locally

IMAP: Copies emails onto server so that multiple clients can access them. Client retrieves emails over TCP/IP connection

On the application layer of TCP/IP model.

DHCP SERVER: Dynamic Host Configuration Protocol

DHCP allocates unique IP address to clients in a NW.

Static Alloc^{n}: DHCP Server **passes** address assigned by network administrator to client

Auto Alloc^{n}: DHCP Server **assigns permanent** IP address to client from its pool of IP addresses.

Dynamic Alloc^{n}: DHCP Server **leases** a **reusable** IP to client for a period of time before it expires. (most often used)

On the network layer of TCP/IP model.

DHCP 4 STEP LEASING PROCESS

Discover: Client **broadcasts** to look for DHCP Server

Offer: DHCP Server offers an IP address

Request: Client requests for server to release address

ACK: DHCP server sends the address to client for ACK

DNS SERVER: Domain Name System

DNS translate domain names to IP addresses. E.g google.com -> 172.217.7.3

Hosts can query the database to be pointed towards the necessary IP address. It is implemented in a hierarchy.

E.g -> ROOT --> Top Level Domain (.com) ---> authoritative (google.com) ----> subdomain (www.google.com)

DATA VALIDATION: Process to ensure that input data is reasonable / correct.

1. Range Check – [1 <= month <= 12]
2. Format Check – date in format [dd/mm/yyyy]
3. Length Check – [password length > 10]
4. Presence Check – [username cannot be blank]
5. Check digit – Check digit is redundant digit or letter calculated from digits of a code number. Permits the accuracy of other digits to be checked. [NRIC last digit]

DATA VERIFICATION: Process to ensure that input data matches the original resource. For example, users are asked to enter the password twice in order to change it – double entry of data is a form of data verification

COMPUTATIONAL THINKING:

Decomposition: breaking down a complex problem or system into smaller, more manageable parts
Pattern Recog.: look for similarities within problems
Abstraction: focus on impt info, ignore irrelevant detail
Algorithms: developing a step-by-step solution to the problem, or the rules to follow to solve the problem

ETHICS:

Integrity: Act with complete discretion when entrusted with confidential information.
Responsibility: Carry out full responsibility in a professional manner, adhere to guidelines.
Competency: Innovate new technology or from other countries. Unethical example: claiming programming proficiency in program language one has not used.

The company's handbook must include rules and regulations for IT staff. Suggest 2 code of conduct for the company. Staff must keep themselves updated with new technologies to protect the digital security of company resources. Staff must protect consumers' data and not use their access rights to modify data.

DATA PRIVACY: The requirement for data to be accessed by or disclosed to authorised persons only.
CONSUMERS: More control, easier access and correction of data, reduced unsolicited telemarketing messages.
BUSINESSES: Increased consumer confidence, Easier cross-border transfer, enhanced efficiency, branding & competitiveness.

PDPA (PERSONAL DATA PRIVACY ACT): Governs the **[1]** collection , **[2]** use and **[3]** disclosure of **[4]** personal data [PD] by organizations in a manner that recognizes both the right of individuals and the need of organizations to collect, use or disclose **[5]** PD for purposes that a reasonable persons would consider appropriate in the circumstances.

PURPOSES OF PDPA

PDPA takes into account – consent, notification, appropriateness, accountability.

1. Collection: actions through which org. obtains PD
2. Use: actions through which org. uses PD
3. Disclosure: actions through which org discloses/ transfers/ makes available any PD.
4. PD: data of an individual who can be identified from that data.
5. Purpose: objective/ reasons relating to PD use.

DATA OBLIGATIONS

1. CONSENT OBLIGATION – Only disclose PD for which an indiv. who has given his/ her consent
2. PURPOSE LIMITATION OBLIGATION – may collect data about indiv. for purposes a reasonable person would consider appropriate + indiv. Has given consent
3. NOTIFICATION OBLIGATION – notify indiv. of purposes for which the org is intending to collect/ use data on / before collection
4. ACCESS & CORRECTION OBLIGATION – PD of an indiv. and info about how it has been used/ disclosed within a year before the request should be provided. However, orgs are not allowed to give access if info could reasonably be expected to cause harmful effect. Orgs are also required to correct any error in PD
5. ACCURACY OBLIGATION – ensure data is accurate
6. PROTECTION OBLIGATION – make reasonable security arrangements to protect the PD that your orgs possesses or controls to prevent unauthorised access, collection, use, disclosure…
7. RETENTION LIMITATION OBLIGATION – cease retention of PD or remove the means why which data can be associated if data is not necessary for any business or legal purposes
8. TRANSFER LIMITATION OBLIGATION – Transfer PDPA to another country only according to requirements stated by regulations, ensure that the standard of protection of data is the same as PDPA.
9. ACCOUNTABILITY OBLIGATION – Make information about data protection policies, practices available on request. Designate a Data Protection Officer to ensure organization complies with PDPA guidelines.

DO NOT CALL REGISTRY (DNC): One can register in the registry to prevent oneself from getting unnecessary marketing calls as PDPA prohibits organisations from sending marketing messages to SG telephone numbers that are registered with the DNC.

3 DNC REGISTERS: No Voice Call/ No Text Message/ No Fax Message.

NRIC (SG NATIONAL REGISTRATION IDENTIFICATION CARD): It is a unique identifier assigned to SG citizens and permanent residents. They are permanent, irreplaceable and used in a variety of government transactions.

COLLECTION OF NRIC: Organizations (excluding governments) are only allowed when NRIC is required under the law or when NRIC is necessary to accurately establish / verify the identities of the individuals (e.g in a clinic). Instead of NRIC, Participants can be identified with their mobile number, or be asked to give the last 4 characters of NRIC for verification purposes.

BACKUPS AND ARCHIVES

BACKUPS: Copy of data that can be used to restore the original if data lost. Original data is not deleted, but older backups are deleted in favour of newer backups. Made on a regular basis. Restore speed crucial. Meant for short term retention. Duplicates periodically overwritten.

ARCHIVE: Copy of data made for *long-term storage,* archive is usually the only copy of data, storing unchanging data that is no longer in use. Retrieval speed not crucial, meant for long term retention. Data cannot be altered or deleted.

VERSION CONTROL: *Easy collaboration across distributed teams*, easy to share code changes & coordinate work. Allows for *better visibility of development pipeline* (overview of what work is going on/ in progress/ who). Maintains change histories, has issue tracking. *Ability to rollback or retrieve previous versions* (because it is incremental). More readily demonstrate compliance and auditing. Allows the maintenance of an audit trail to account for all changes and details those with access to it.

NAMING CONVENTION: Code is written once but read multiple times, hence readability is impt. Naming conventions are names that everyone agrees to follow, improves communication and is easier to read files. E.g: UPPERCASE, camelCase, PascalCase. Words can be separated by case, underscores, or spaces.

PROGRAMMING ELEMENTS

TOP-DOWN DESIGN: Method of designing a solution to a problem by splitting it up into simpler problems.

MODULARITY: is the feature of designing problem solutions as a collection of well-defined separate modules.

+: Modules can be coded and tested separately, ease of debugging (each module coded & tested separately), easier to maintain small modules, can be used in other programs, easier to monitor and control large projects.

-: Larger memory space, more files, time needed to decompose problem into module.

State what is meant by a recursive function and when it is suitable to be used. Give one advantage and disadvantage of using recursive functions.

RECURSIVE FUNCTION is a function defined in terms of itself, that can repeat itself several times until one or more terminal case(s) is reached. Used when original task can be reduced to a simpler version of itself.

+: Simplifies the design of algorithms and code.

-: Requires more memory than iterative.

PROGRAM ERRORS:

Syntax Error: Mistakes in the source code.

Semantic / Logic Error: Errors in a program that causes it to operate incorrectly, but not to terminate abnormally. (Produces unintended / undesired output)

Runtime Error: Errors that occurs while the program is running after being successfully compiled.

ABSTRACT DATA TYPES

ADT are a class of objects whose logical behaviour is defined by a set of values and a set of operations.

STATIC VS DYNAMIC MEMORY ALLOCATION:

Static memory allocation is determined at compile time. Once it is allocated, it cannot be changed. (e.g Array) Dynamic memory allocation is done during program execution. When memory is allocated, the memory size can change (e.g Linked Structure)

ASCII AND UNICODE

ASCII is a character encoding standard for electronic communication *(only English)*. Unicode is a universal character encoding standard that assigns a code to *every character and symbol in every language* in the world.

DATABASE (DB): Organized collection of structured info

DATA REDUNDANCY: Same data stored more than once

DATA INTEGRITY: Data should not be entered twice (accuracy), needs to change everywhere (consistency)

FILE - BASED (FB) APPROACH: Sharing file (server/ email..)

-: Lack of privacy; data not suitable; hard to update; data dependency issues (compatibility btw software)

DB APPROACH: External, Conceptual, Internal Levels

+: Multiple (user) views; Won't affect other user; Insulation between program & data; multiuse transaction processing, program-data independence.

DB TYPES: Relational (tables, columns, rows); NoSQL (non-relational); Flat-file; hierarchical; network.

DB vs FB: Atomicity – changes are performed as single operation | **C**onsistency – data in consistent state | **I**solation – Intermediate state of transaction is invisible | **D**urability – Changes to data persists, even if sys failure.

DBMS: Interface between database and end users or programs, allowing users to retrieve, update, manage how the info is organized, and optimized.

-: High initial investment (additional hardware, manpower); When not necc; When Data is too complex

RELATIONAL DATA MODEL: Organizes & represents data in form of a table. (Based on math. concept of relation)

RELATION (TABLE): Table with rows and columns

TUPLE (ROW): Row of a relation

ATTRIBUTE (COLUMN): Named column of a relation

DOMAIN: Set of allowable values for attribute

CANDIDATE KEY: Minimal attributes that identify tuples

PRIMARY KEY (PK): Candidate key to identify tuples UNIQUELY. | Criteria: less likely to have its values changed/ lose uniqueness; with fewest characters; min. set of attributes, easier to use from user's perspective.

SECONDARY KEY: Candidate key that are not PK.

FOREIGN KEY: Attribute that provides logic link btw tables.

COMPOSITE KEY: Combination of two or more attributes.

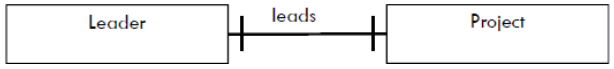
ENTITY RELATIONSHIP DATA (ERD) MODEL: Used for data modelling for databases (easy to discuss and explain)

ENTITIY: Tables that hold specific info (rep by rectangles)

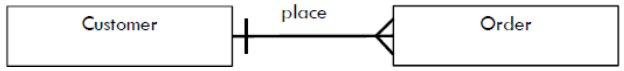
RELATIONSHIPS: Assoc. / interac^{n} btw entities (rep by line)

CONNECTIVITY: One to One; One to Many; Many to Many

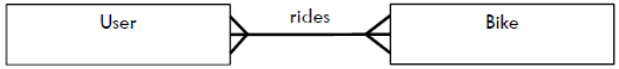
1-1: Each project 1 leader | Each leader leads 1 project



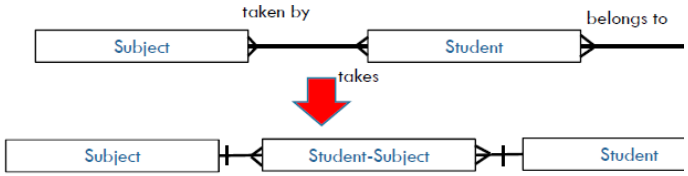
1-Many: User place many orders | Each order 1 user.



Many-Many: User got many bikes | each bike many users.



REFINEMENT: (Split the M:M R/S into 2x of 1:M R/S)



DATA REDUNDANCY (DR): When data is being stored more than once. Causes data anomalies.

DATA DEPENDENCIES: A dependency is a constraint that applies to or defines the r/s btw attributes. Occurs when info stored in the same database table uniquely determines other information stored in the same table.

DATA ANOMALIES: When not all changes successful

1. Insertion anomaly: Until class created, student data cannot be inserted, unless it is set as NULL
2. Update anomaly: If a class name is changed, all records need to be updated, else there is data inconsistency
3. Deletion anomaly: If student table has student and class info, when table deleted, class info lost too

FUNCTIONAL DEPENDENCIES (FD): Eliminate redundancy & anomalies; ensures all attributes in a table belong there. If you derive Y from X, Y is functionally dependent on X.

NORMALIZATION: New DB: Analyse R/S among attributes in ERD to improve. | Existing DB: Analyse R/S among attributes in data structure to improve DB structure

1st NORMAL FORM (NF): Each attribute has ONE value.

STUDENTID	NAME	SUBJECT	
~~0001~~	~~Daren~~	~~MA, CP~~	NOT 1NF
0001	Daren	MA	1NF
0001	Daren	CP	

2ND NF: 1NF + All non-key attributes FD on (composite) PK.

SCOREID	STUDENTID	MARKS	SUBJ_ID	TEACHER
1	0001	70	1	Ms MA
2	0001	30	2	Mr CP
3	0002	70	1	Ms MA

** In this case, STUDENTID & SUBJ_ID are composite PK. TEACHER is only dependent on SUBJ_ID. Hence, it is not FD on composite PK. To remove **partial dependency**, create new SUBJ_ID table (SUBJ_ID, TEACHER NAME)

STUDENT TABLE

SCOREID	STUDENTID	MARKS	SUBJ_ID (FK)
1	0001	70	1
2	0001	30	2
3	0002	70	1

SUBJECT TABLE

SUBJ_ID	TEACHER_NAME
1	Ms MA
2	Mr CP

3RD NF: 2NF + Not Transitive Dependent (TD).

SCOREID	STU_ID	SUBJ_ID	MARKS	EXAM_NAME	TOTALMARK
1	0001	1	70	Lab	100
2	0002	2	30	Theory	50

TD: SCOREID -> EXAM_NAME -> TOTALMARK

SCOREID -> TOTALMARK is a TD

Put EXAM_NAME, and TOTALMARK into another table with PK EXAM_ID, use EXAM_ID as Foreign Key in SCORE.

STUDENT TABLE

SCOREID	STU_ID	SUBJ_ID	MARKS	EXAM_ID(FK)
1	0001	1	70	9931
2	0002	2	30	9922

EXAM TABLE

EXAM_ID(PK)	EXAM_NAME	TOTAL_MARK
9931	Lab	100
9922	Theory	50

SQL: Create database + structures | Basic data mgmt. | Complex queries to change data into useful info

DATA DEFINITION LANGUAGE: Define + Create DB (objs)

1. CREATE: Create DB| For CREATE TABLE, check below
   `CREATE DATABASE` *`database-name;`*
2. ALTER: Alter DB structure
   ```
   ALTER TABLE Classes
   RENAME TO Class
   ADD ClassID INTEGER (note: run once at a time)
   ```
3. TRUNCATE: Remove ALL records
   `DELETE FROM` *`table-name;`*
4. DROP: Delete DB, Table, Objects
   `DROP TABLE [IF EXIST]` *`table-name;`*

```
FOREIGN KEY CONSTRAINT:
CREATE TABLE STUDENT (
StudentID INTEGER PRIMARY KEY AUTOINCREMENT,
ClassID INTEGER,
Name TEXT(100) NOT NULL,
NRIC TEXT(15) NOT NULL,
FOREIGN KEY(ClassID) REFERENCES
Class(ClassID) );
```

DATA MANIPULATION LANGAUGE:

1. SELECT
   SELECT *expressions* FROM *table* WHERE *conditions*
   Use * as expression to select ALL | Conditions can be like '%TAN%' | %: returns matches of >= 0 chars | "BETWEEN 5 and 20" | BETWEEN is inclusive of 5 & 20

```
SELECT Todo.id, Category.name, Todo.description
FROM Todo INNER JOIN Category ON
Todo.categoryName = Category.id
```

2. INSERT
   INSERT INTO *table*(columns) VALUES(values list)

```
INSERT INTO Category(Name,Description)
VALUES('Stationary','Writing Materials')
```

3. UPDATE
   UPDATE *table* SET *column_name = column_value ..*
   WHERE *conditions*

```
UPDATE Category SET Name='Stationary' WHERE id>5
```

4. DELETE
   DELETE FROM *table* WHERE *conditions*

```
DELETE FROM Category WHERE id=5
```

Examples: 'SQL ALL COMMANDS.sql', 'SQL Exercise 1'

SQLITE3 (PYTHON)

PYTHON SQL SELECT FROM TABLE

```
import sqlite3
db = sqlite3.connect('database.db')
db.row_factory = sqlite3.Row
cursor = db.cursor()
cursor.execute("SELECT * FROM Category")
category = cursor.fetchall()
```

NOTE: Without line 3, what returned is *a list of tuples of the column data* within. With line3, what returned is a *list of dictionaries with key column_name and value column_value* || ** SQL CREATE STATEMENT:

```
CREATE DATABASE database.db
```

PYTHON SQL INSERT INTO TABLE

```
db.execute("""INSERT INTO
Todo(categoryName, description)
VALUES (?,?);""", (category,description))
db.commit() #this is important
```

NOTE: Using triple ""s, we can escape all brackets within query. | The '?'s mark variables within python that we want to insert into the query. | Notice the brackets, and that python variables are stored in a tuple.

MONGODB (NOSQL):

WHY: ~ is a non-relational database (no predefined schema). | Document database | Horizontally scalable. **+:** Dynamic schema, Fast data storage, Handle large data **WHY SQL:** Data has fixed schema | Complex & Varied queries frequently performed | Atomicity, consistency, isolation, durability (ACID) is impt | Supports high number of simultaneous clients

```
from pymongo import MongoClient
client = MongoClient('localhost',27017)
mydb = client['database']
mycoll = mydb['customers']
bookList = [{'bookid':234},{'bookid':331}]
x = mycoll.insert_many(bookList)
print(x.inserted_ids)

query = {'bookid':{"$gte":100, "$lte":200}}
q2 = {'desc':{'$regex':'hi'}}
q3 = {'desc':'hi'}
q4 = {'$or':[{'borough':"Staten"}, {'borough':"Queens"}]}
q5 = {'borough':{'$in':["Staten", "Queens"]}}
q6 = {"address.coord":{$type: 1}}
number = coll.find_one(query).get('freq') #freq dict value
count = coll.count(query)

projection = {"_id":0}
result = mycoll.find(query, projection).sort('bookid',1)

mydb.collection.update_many({"bookid":{"$exists":False}},
{"$set":{"bookid":000}})
```

regex query has to be in dict | address.coord – dict of values within another dict | .sort('borough',1) or -1 diff btw q2 & q3: q3 is exact, q2 ignores trailing char **common regex queries:** $eq,$ne,$gte,$gt,$lte,$lt, $in(q5),$nin| $and,$not,$or(q4)| $exists| $set, $unset| $type:1/2/3/4(double/str/obj/array) (q6)

SOCKET

```
# THIS IS CLIENT CODE
import socket
s = socket.socket()
s.connect(('localhost',9999))
data = b''
send = ''
while send != 'end':
    send = input('Input CLIENT -> ')
    s.sendall(send.encode())
    data = s.recv(1024)
    print(f"Received from SERVER: {data.decode()}")
s.close()
```

```
# THIS IS SERVER CODE
import socket
s = socket.socket()
s.bind(('localhost',9999))
s.listen()
s, address = s.accept()
data = b''
send = ''

while True:
    data = s.recv(1024)
    print(f"From connected user: {data.decode()}")
    if data == b'end':
        break
    send = input("Input SERVER -> ")
    s.sendall(send.encode())

send = "'Ok, then. bye!'"
print("INPUT SERVER ->", send)

s.sendall(send.encode())
s.close()
```

FLASK:

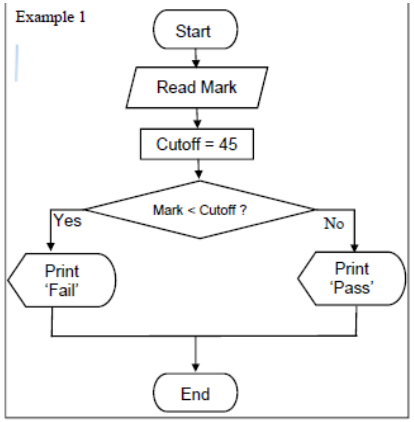
```
import flask, sqlite3, os
from flask import *
from werkzeug.utils import secure_filename
app = flask.Flask(__name__,
template_folder="TASK4_3")

@app.route('/', methods=['GET','POST'])
def home():
    if request.method == 'GET':
        return render_template('home.html')
    else:
        name = request.form['name']
        temp = request.form['temp']
        date = request.form['date']
        try:
            photo = request.files['photo']
              filename=secure_filename(photo.filename)
              imageURL= os.path.join('uploads', filename)
               photo.save(imageU RL)
        except:
            filename = None
     render_template('student.html', name=name)

@app.route('/uploads/<filename>')
def getFile(filename):
    return send_from_directory('uploads', filename)
app.run()
```

FLOWCHARTS

A flowchart is a formalized graphic representation of a logic sequence, which provides people with a common language or reference point when dealing with a project or process.



Symbol	Symbol Name (alias)	Symbol Description
Process / Operation Symbols		
	Process	Show a Process or action step. This is the most common symbol in both process flowcharts and business process maps.
Branching and Control of Flow Symbols		
	Flow Line (Arrow, Connector)	Flow line connectors show the direction that the process flows.
	Terminator (Terminal Point, Oval)	Terminators show the start and stop points in a process. When used as a Start symbol, terminators depict a *trigger action* that sets the process flow into motion.
	Decision	Indicates a question or branch in the process flow. Typically, a Decision flowchart shape is used when there are 2 options (Yes/No etc.)
Input and Output Symbols		
	Data (I/O)	The Data flowchart shape indicates inputs to and outputs from a process. As such, the shape is more often referred to as an I/O shape than a Data shape.
	Display	Indicates a process step where information is displayed to a person (e.g., PC user, machine operator).

PSEUDOCODE

DECLARE: Creation of variable in memory
CONSTANT: Special variable that cannot be changed.
ASSIGNMENT: <identifier> **<--** <value>
INPUT <identifier>: (collect user input)
OUTPUT <values>: displays values

IF <condition>
  THEN
    <statement>
  ELSE
    <statement>
ENDIF

FOR INDEX = 1 TO 5:
    Extract <- Extract + Sentence[Index]
ENDFOR

WHILE <condition> DO
  <statement>
ENDWHILE

PROCEDURE <identifier>
  <statement>
ENDPROCEDURE

FUNCTION <identifier> RETURNS <data type>
  <statement>
  RETURN <variable>
ENDFUNCTION

DECISION TABLES

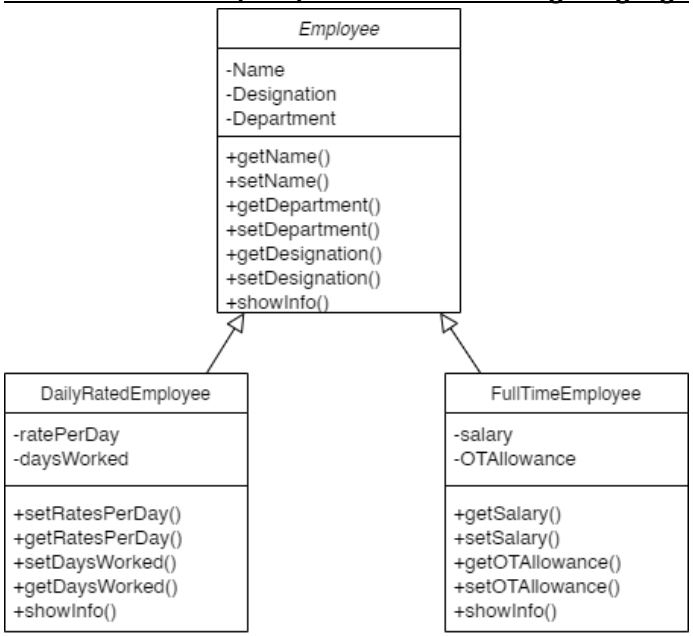
Rules to decide whether to offer insurance to customers and offer discounts are as follows: *[HCI 2018 PRELIM]*

- If the customer has been refused insurance by another company and their car is over 10 years old then insurance is refused.
- If the customer has been refused insurance by another company and their car is not more than 10 years old then insurance without any discount is available.
- If the customer has not been refused insurance by another company and their car is over 10 years old then insurance without any discount is available.
- If the customer has not been refused insurance by another company and their car is less than 10 years old and they have made not more than three claims previously then insurance with a discount is available.

Refused by other company	Y	Y	Y	Y	N	N	N	N
Car > 10 years old	Y	Y	N	N	Y	Y	N	N
Claims > 3	Y	N	Y	N	Y	N	Y	N
Insurance Refused	X	X						
Insurance w/o discount			X	X	X	X	X	
Insurance with discount								X

Refused by other company	Y	Y	N	N	N
Car > 10 years old	Y	N	Y	N	N
Claims > 3	-	-	-	Y	N
Insurance Refused	X				
Insurance w/o discount		X	X	X	
Insurance with discount					X

CLASS DIAGRAMS (ADT) – Unified Modelling Language



Take note: The direction of arrows, and to write "+" and "-" for public and private methods/ attributes

Polymorphism refers to the ability for code to take different forms. E.g: `showInfo()` function is overridden in the subclasses, although it is defined in superclass.

Code reuse is seen through the inheritance of the employee class by subclasses. Methods of employee class are inherited without any coding -> code reuse.

Classes are code that specifies the data attributes and methods of a particular type of object

Instances are objects created from a class.

Encapsulation: combining methods & attributes into a single object.

Data Hiding: Object's data attributes are hidden from code outside object, access is restricted to it' s methods.

Advantages and Disadvantages of Polymorphism: Code reuse, and reduces the coupling between different functionalities but difficult to implement and poor runtime and readability

TRACE DIAGRAMS

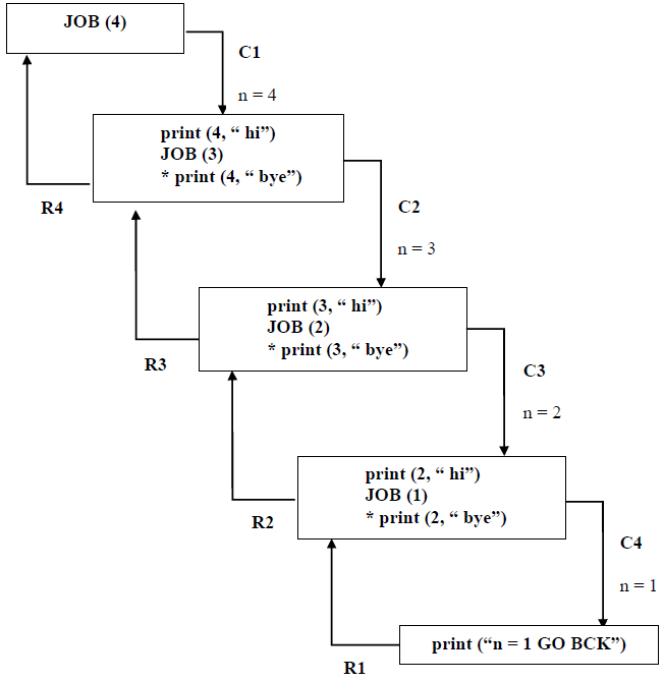
NOTE: I was unable to find this in any block test/ prelim papers for the past few years.

Trace diagrams are used to visualise how recursive functions work.

CODE:	OUTPUT:

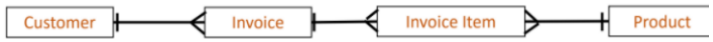
DEF JOB(n):
  if n == 1:
  print('n = 1 GO BCK')
  else:
  print(n, 'hi')
  JOB(n - 1)
  print(n, 'bye')
	4 hi
	3 hi
	2 hi
	n = 1 GO BCK
	2 bye
	3 bye
	4 bye

C: Call | R: Return



ERD (Entity Relation Diagram)

NOTE: How the relations are drawn is on page 4.



Customer(CustomerID, Name, Phone, Email)
Invoice(InvoiceID, InvoiceDate, Subtotal)
InvoiceItem(ItemID, InvoiceID, ProductId, Date, Qty, Total)
Product(ProductID, Description, UnitPrice)

NOTE: InvoiceItem has ItemID as primary key, and InvoiceID, ProductID as foreign key.

RUNTIME STACK

```
def recur(string):
    if len(string) == 0:
        return ''   # empty string
    else:
        return recur(string[1:]) + string[0] * 2
```

Use diagrams wherever possible to show:

- the order in which the calls to the function are made
- the order in which the returns are made
- the data that are stacked at each call.

[5]

Write down the value of the above function call `recur('HCI')`. [1]

Solution:

- An **activation record** is created to store the **parameters,** variables**, return address** and return value of a function.
- When a function is **called**, we **push** its activation record onto the run-time stack.
- When a function is **terminated**, we **pop** the activation record from the run-time stack.
- The **top** of the run-time stack is always the function that's currently executed.

After first four calls:

4th call recur('')	''	?	A
3rd call recur('I')	'I'	?	A
2nd call recur('CI')	'CI'	?	A
1st call recur('HCI')	'HCI'	?	B
	string	return value	return address

1st return of recur(''):

	''	''	A

3rd call recur('I')	'I'	?	A
2nd call recur('CI')	'CI'	?	A
1st call recur('HCI')	'HCI'	?	B
	string	return value	return address

2nd return of recur('I'):

	'I'	'II'	A

2nd call recur('CI')	'CI'	?	A
1st call recur('HCI')	'HCI'	?	B
	string	return value	return address

3rd return of recur('CI'):

	'CI'	'IICC'	A

1st call recur('HCI')	'HCI'	?	B
	string	return value	return address

4th return of recur('HCI'):

	'HCI'	'IICCHH'	B

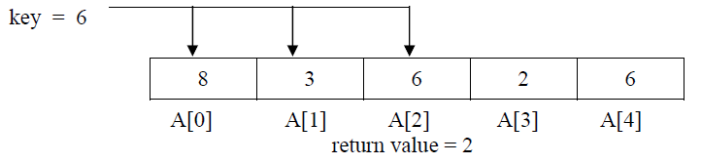
Empty Stack

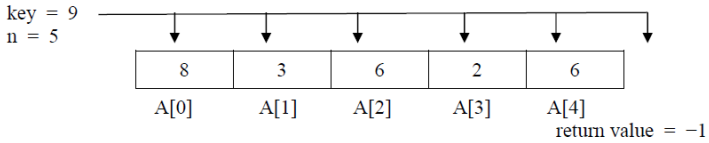
string	return value	return address

SEARCHING

LINEAR SEARCH

BEST: O(1) | AVG: O(n) | WORST: O(n)
BEST: key == A[0] | WORST: key == last element

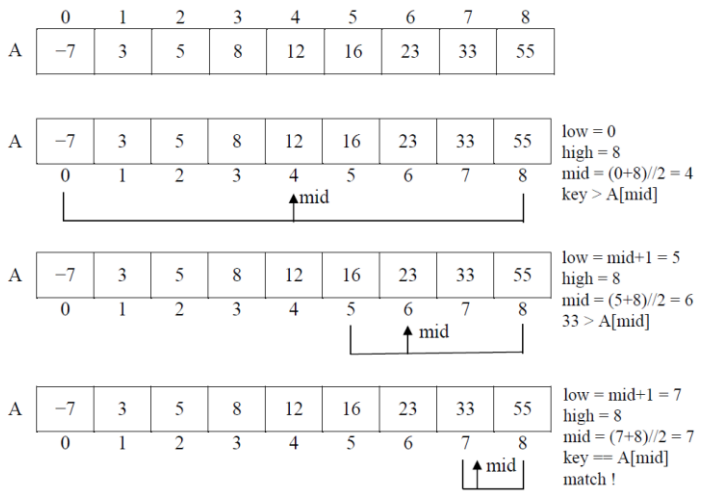




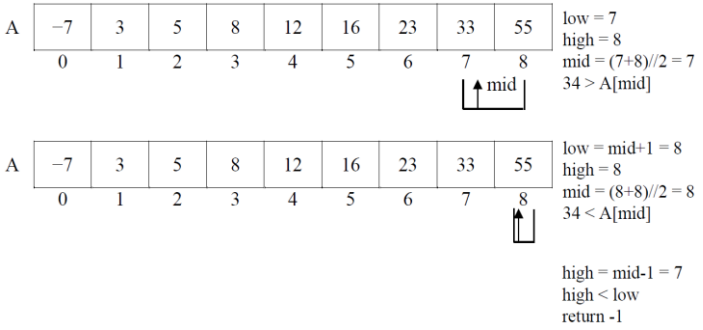
BINARY SEARCH

BEST: O(1) | AVG: O(log n) | WORST: O(log n)
BEST: key==A[mid] | WORST: extreme/ not in list

When key = 33:



If key = 34, instead of the last step:



HASH TABLE SEARCH

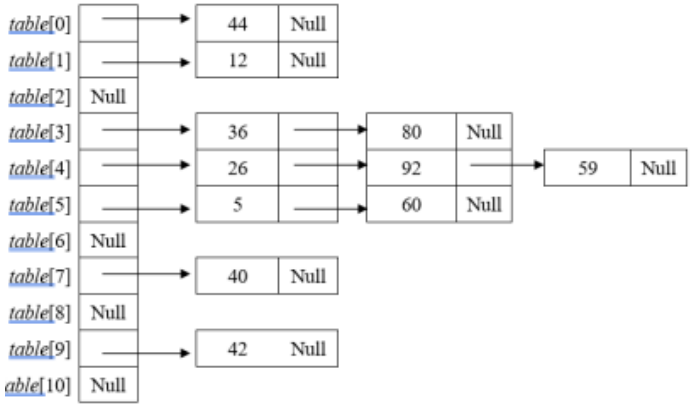
BEST: O(1) | AVG: O(1) | WORST: O(n)
BEST: Chaining & Good hash function.
WORST: Too many elements hashed into same key

ARRAY = [26, 42, 5, 44, 92, 59, 40, 36, 12, 60, 80]

Linear Probing

0	1	2	3	4	5	6	7	8	9	10	<- Index
				26							26 % 11 = 4
				26					**42**		42 % 11 = 9
				26	**5**				42		5 % 11 = 5
44				26	5				42		44 % 11 = 0
44				26	5	**92**			42		92 % 11 = 4
44				26	5	92	**59**		42		59 % 11 = 4
44				26	5	92	59	**40**	42		40 % 11 = 7
44			**36**	26	5	92	59	40	42		36 % 11 = 3
44	**12**		36	26	5	92	59	40	42		12 % 11 = 1
44	12	**60**	36	26	5	92	59	40	42		60 % 11 = 1
44	12	60	36	26	5	92	59	40	42	**80**	80 % 11 = 1
44	12	60	36	26	5	92	59	40	42	80	FINAL

Chaining



SORTING

NOTE: All searching and sorting code in separate jupyter notebook.

BUBBLE SORT

BEST: O(n) | AVG: O(n^2) | WORST: O(n^2)
BEST: Already sorted | WORST: Sorted opposite

Pass 0:

A[0]	A[1]	A[2]	A[3]	A[4]	
50,	20,	40,	75,	35	Exchange 50 and 20
20,	50,	40,	75,	35	Exchange 50 and 40
20,	40,	50,	75,	35	50 and 75 are ordered
20,	40,	50,	75,	35	Exchange 75 and 35
20,	40,	50,	35,	**75**	75 is the largest element lastExchangeIndex = 3

Pass 1:

A[0]	A[1]	A[2]	A[3]	A[4]	
20,	40,	50,	35,	**75**	20 and 40 are ordered
20,	40,	50,	35,	**75**	40 and 50 are ordered
20,	40,	50,	35,	**75**	Exchange 50 and 35
20,	40,	35,	**50**,	**75**	50 is the largest element lastExchangeIndex = 2

Pass 2:

A[0]	A[1]	A[2]	A[3]	A[4]	
20,	40,	35,	**50**,	**75**	20 and 40 are ordered
20,	40,	35,	**50**,	**75**	Exchange 40 and 35
20,	35,	**40**,	**50**,	**75**	40 is the largest element lastExchangeIndex = 1

Pass 3:

A[0]	A[1]	A[2]	A[3]	A[4]	
20,	35,	**40**,	**50**,	**75**	20 and 35 are ordered
20,	**35**,	**40**,	**50**,	**75**	Ordered List lastExchangeIndex = 0

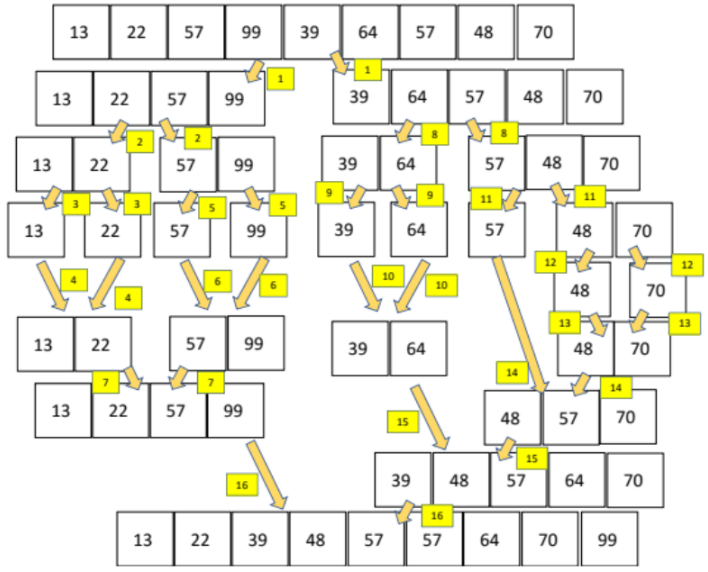
INSERTION SORT (Note: came out twice before)

BEST: O(n) | AVG: O(n^2) | WORST: O(n^2)
BEST: Already sorted | WORST: Sorted opposite

input list:	GRA	CHR	DAV	SAR	TOM	KAT
start with GRA:	**GRA**					
insert CHR:	**CHR**	GRA				
insert DAV:	CHR	**DAV**	GRA			
insert SAR:	CHR	DAV	GRA	**SAR**		
insert TOM:	CHR	DAV	GRA	SAR	**TOM**	
insert KAT:	CHR	DAV	GRA	**KAT**	SAR	TOM
sorted list:	CHR	DAV	GRA	KAT	SAR	TOM

MERGE SORT

BEST/WORST/AVERAGE CASE: O(n $\log_2(n)$)
SAME: Split array into 2& need time to merge back



QUICKSORT

BEST: $O(n\text{Log}_2(n))$ |AVG: $O(n\text{Log}_2(n))$ |WORST: $O(n^2)$
WORST: Already Sorted

[HCI 2019 PRELIM]: Show how the numbers can be sorted in ascending order using a quicksort. For each pass, show the numbers swapped and the sub lists after splitting.

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	
435	646	344	54	23	98	Original list
344		**435**				low = 0, high = 5, pivot = A[2] =344
	98				**646**	left = 1, right = 5, swap 646, 98
		23		**435**		left = 2, right = 4, swap 435,23
						left = 4, right = 3, pos = 3
54	98	23	344	435	646	Swap 344, 54. Partition Done.
54	98	23				**Left sublist**
98	**54**					low = 0, high = 2, pivot = A[1] = 98
						left = 3, right = 2, pos = 2
23	54	98				Swap 98, 23. Partition Done.
23	54					**Left sublist**
						2-element sublist requires no partition. Compare 23 & 54, no swap required
				435	646	**Right sublist**
						2-element sublist requires no partition. Compare 435 & 646, no swap required
23	54	98	344	435	646	Sorting is done.

WEB APPLICATION THEORY

Web Application is an application stored at a remote server that requires a web browser as a client to run and internet for data/ resource transfer.

Differences between web apps and native applications:

Web App	Native Apps
Deployment and any update requires servers	Deployment and any update are done on individual client machines
No location constraint, accessable anywhere	Can only be accessed from the machines they are deployed in
Higher security risks (due to the accessibility)	More secure, since better access control by admins
Platform-independent. Only requires web browser	Desktop apps need to be developed separely for diff platforms
Heavily reliant on internet connectivity	Does not require internet. Only for updates.

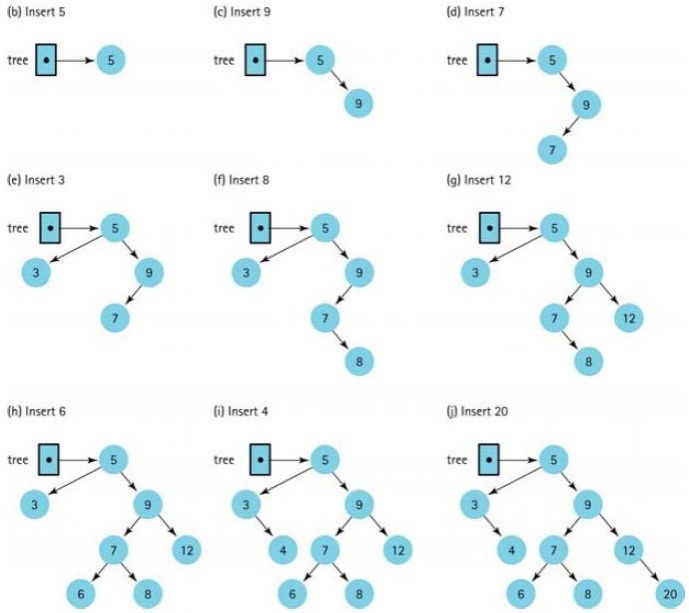
Usability principles in design of web apps

Visibility of System Status	Loading bars keep users informed about what is going on
Match between system and real world	System should speak the users' language, rather than system-oriented terms.
User control and freedom	Supporting undo and redo. Allowing users to easily remove attachments on email
Consistency and standards	Platform conventions e.g having the currency choices and login options on top right hand corner.
Error Prevention	Google offers the correct word when user inputs wrong spelling, Gmail stops users from sending email without subject or text.
Recognition rather than recall	Minimize the user's memory load. E.g showing past user search history / Google autocomplete
Flexibility and efficiency of use	Allowing systems to carter to both inexperienced and experienced users. E.g Google Advanced Search Engine
Aesthetic and minimalist design	Only information that is relevant and necessary is shown.
Helping users recognise, diagnose, recover from errors	Error messages are expressed in plain language, precisely indicate the problem, and suggest a solution. E.g – sign in auth errors will suggest 'forgot my password'
Help and documentation	Ideally, systems do not need documentation. But it should be easy to search, consise and not too large.

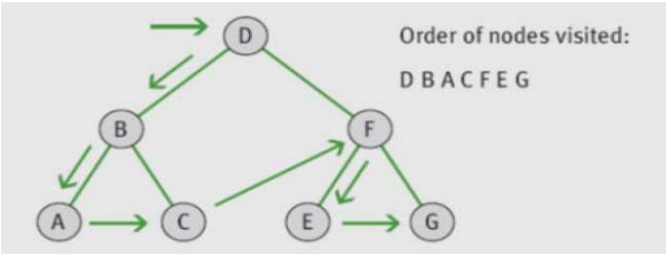
DEBUGGING

1. Whitebox testing
2. Trace diagram

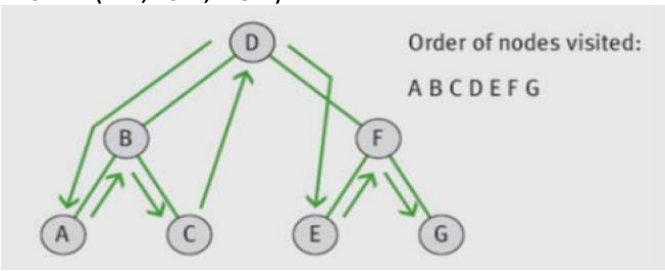
BINARY SEARCH TREE



PRE-ORDER (NODE, LEFT, RIGHT):



IN-ORDER (LEFT, NODE, RIGHT):



POST-ORDER (LEFT, RIGHT, NODE):