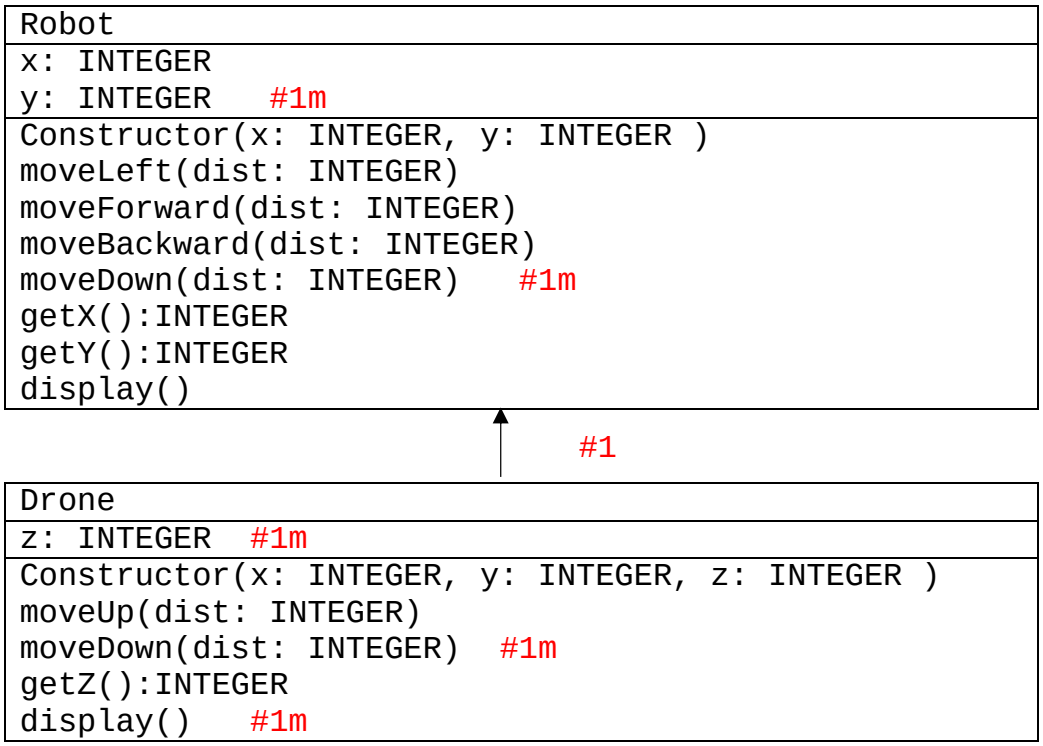
1a OOP 6m



Each mistake (repeated/missing attributes or methods, missing arrow) minus 1 m

1b **i. Encapsulation**

Encapsulation is used to hide the values/attributes of an object inside a class, preventing unauthorized parties' direct access to them. Publicly accessible methods are provided in the class to access the values. [1m]

Example, attributes x, y, z are private, getX, getY, getZ are publicly accessible [1m]

ii. Inheritance

In inheritance, the child class inherits the attributes and methods from the parent class. [1m]

Examples of inheritance, x and y attributes, moveLeft, moveRight, etc… methods. [1m]

iii. Polymorphic

Polymorphism in python means methods in the child class that have the same name as the methods in the parent class.
The method in the child class overwrites/modifies the code that it has inherited from the parent class. [1m]

Display method is polymorphic. [1m] 6m

2 a) Role and Purpose of a Primary Key

i. A primary key is a unique identifier for each record in a table. [1] It helps enforce entity integrity, ensuring that there are no duplicate records in the table. [1]

In the context of the "GreenHarvest" scenario, the primary key ensures that each farmer or crop has a unique identifier (farmerID or cropID), preventing data duplication.

ii. A foreign key is a field in one table that establishes a link between the data in two tables. [1] Its role is to maintain referential integrity, ensuring that data in one table corresponds to data in another table. [1]

In the "GreenHarvest" scenario, foreign keys in the HarvestRecord table link each harvest to a specific farmer and crop (farmerID and cropID), ensuring that each harvest record relates to a valid farmer and crop in their respective tables.

b) In the context of "GreenHarvest," data redundancy refers to the repetition of the same data within the database. [1] For example, having to store the same crop's details for each harvest record can be redundant. [1]

Data inconsistency, within the context of "GreenHarvest," refers to conflicting or contradictory data values within the database (i.e.: the same piece of information is stored differently in various places). [1] For instance, if the farmer’s name is stored differently in different harvest records, it can create inconsistencies. [1]

c) In the context of "GreenHarvest," unresolved data redundancy can result in update anomalies. For instance, if information about crops, such as planting dates, is stored redundantly across multiple records in the Crop table. [1]

Suppose you need to update the planting date of "Crop A" because it was initially recorded incorrectly. If this information is redundantly stored in multiple harvest records, you must update it in all related records. Failure to do so can lead to inconsistencies, where some records show the corrected date while others display the incorrect one. [1]

d) Table Descriptions:

Farmer (<u>farmerID</u>, FirstName, LastName, ContactNumber)
Crop (<u>cropID</u>, CropName, CropType, PlantingDate)
HarvestRecord (<u>farmerID</u>*, <u>cropID</u>*, <u>HarvestDate</u>)

[3] Farmer with additional 2 attributes and primary key
[3] Crop with additional 2 attributes and primary key
[3] HarvestRecord with composite and foreign key

e) ER diagram **(4 marks instead of 6 marks)**

[1] Three tables
[1] correct sequence
[1] relationship between farmer and harvestrecord
[1] relationship between harvestrecord and crop

3a Binary Tree
Answer:

(i) D B E A F C G 2

(ii) X is In-order traversal 1

b Answer:

```
FUNCTION Count(CurrPtr) RETURNS INTEGER
    IF CurrPtr = -1  # base case                          1
        RETURN 0                                          1
    ELSE
        RETURN 1 + Count(BT(CurrPtr).LeftPtr)             2
                 + Count(BT(CurrPtr).RightPtr)            1
    ENDIF
ENDFUNCTION
```

4a When n = 1, Fib(n-2) would be Fib(-1) [1]
which causes infinite recursion leading to memory stack overflow. [1]

Add the following base case to the pseudocode: [1]

```
IF n = 1 THEN
  RETURN 1
ENDIF
```

4b

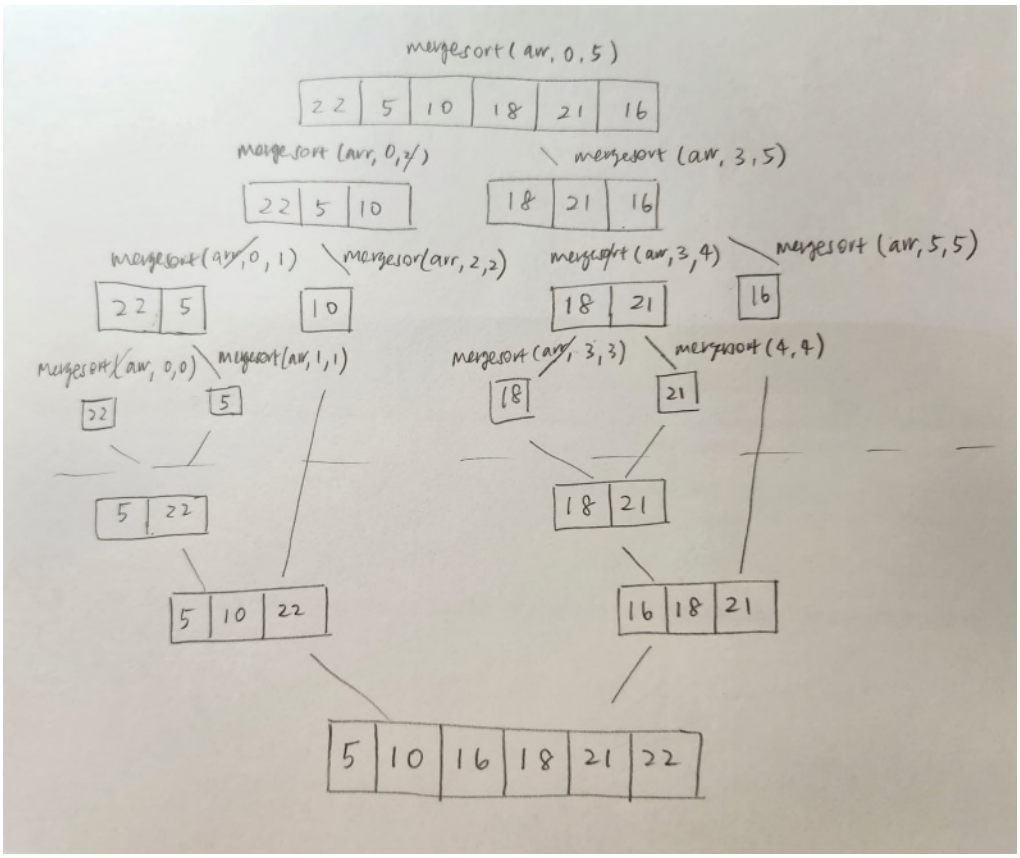
Test case	Data	RETURN	Purpose
1	n = 0	0	Test base case 1
2	n = 1	1	Test base case 2
3	n = 2	1	Test general case

4c **7 marks instead of 5 marks**

Step 1: Return n if n is less than or equal to 1.
Step 2: Declare temp, termA and termB.
Assign 0 to temp, termA and 1 to termB.
Step 3: Add termA to termB and assign the sum to temp.
Step 4: Assign termB to termA and temp to termB.
Step 5: Subtract 1 from n.
Step 6: Repeat from step 4 if n is larger than 0. Otherwise return termA.

Steps 1 to 5 – 1 mark each
Step 6 – 2 marks

5
a,b)



4 marks from the diagram
4 marks for labelling function calls for mergesort
0 m - merge no need to label

5c For bubble sort, the time complexity of a mostly sorted array is close to that of a pre-sorted array, which is **O(n)** due to the use of a boolean flag to check and terminate the sort when no swap was done in a pass. 1

For merge sort, the time complexity is **O(n $\log_2$n)** for pre-sorted array, as well as for best, average and worst case. It is fixed 1

Thus, bubble sort is faster for **mostly sorted array.**

6a Network + Security

(a) Reason for layering

(**Any two advantages)**:

- Simplifies the network model; Enables programmers to specialize in a particular layer of the model;
- Provides design modularity; Allows for standardized interfaces to be produced by networking vendors

1. Easy Debugging
It is very simple to debug because the layers are discrete. Example if an error happens in the application layer, the developer may only debug that layer.

2. Modularity
This design supports modularity because each layer only executes tasks it is responsible to perform.

3. Abstraction
Each layer is concerned with its own set of functions. As a result, the functions and implementations of the other layer are abstract to it.

4. Easy update
A modification in one layer does not affect the other layers.

6b (b) **Transport Layer**: The Transport layer is responsible for providing reliable data transfer between applications. It has two protocols:

1. Transmission Control Protocol (**TCP**) [1]

- is a connection-oriented protocol that establishes a reliable connection between two devices before transferring data. It ensures that all the data is received in the correct order and without errors. [1]
- Useful for online transaction, emails. [1]

6b 2. **User Datagram Protocol** (**UDP)** [1]

- on the other hand, is a connectionless protocol that does not provide error checking or guaranteed delivery. [1]
- Useful for real-time services like live-conference or online games [1]

Any one of the above.

6c The router uses its NAT table to map the arriving packet's destination public IP address and port with the private IP address and port of the device in the LAN, and send the packet to the device. 2

6di
- JPNet users' computers could be infected with Malware sent through email. These infected computers form a botnet. 1
- When the attacker launches an attack, these botnets will work simultaneously to send large amount of network traffic in JPNET. 1
- This will cause JPNET to be flooded with requests, its bandwidth to be used up and thus out of service. 1

ii - Install Intrusion Prevention System (IPS) to monitor the network and take action to prevent any malicious activity by reporting, blocking or dropping it when it occurs. 1

Any other reasonable strategy:

- Scalable bandwidth ;
- Captcha to accept only genuine human request
- Update antivirus software to detect malware installed and clean the computer 1

No marks:

Install Intrusion Detection System (IDS) alone is insufficient because it only monitors the network.

Install Firewall is not acceptable because DDOS entry point is usually an open firewall port.

Firewalls simply block access to certain ports, which hackers can easily work around. Firewalls don't necessarily stop phishing attempts, prevent others from eavesdropping, or stop users from installing malware or falling victim to social engineering schemes

6e

Weight	2	3	4	5	6
Hex	2	2	A	0	1
Product	2 x 2	2 x 3	10 x 4	0 x 5	1 x 6
	4	6	40		6

2

Check digit = (4+6+40+6)%16 = 56 % 16 = 8

2

7a Data Mgmt and Ethics

a) Data breach is exposing confidential, sensitive or protected information to an unauthorized person. The files in a data breached are viewed and/or shared without permission. 1m

b 3m

c

Causes	Measures
Insider:	
An employee using a co-worker's computer, read and copied the files without having the proper authorization permissions.	- Never leave laptop/devices unattended - Do not share login account with colleague - Digital certificate to authenticate user
The insider has **authorization** to access the data, and the intent is to use the information in malicious ways.	- Non disclosure/confidentiality agreement between organization and employee - Access control policy where every subject should be given the minimum set of access rights necessary to complete the job.
Outside criminals:	
Hackers have planted a **malware** in a staff computer through email downloads. Through the intranet, the malware-infected computer stole the customers' information from the database.	Intrusion Prevention System (IPS) to continuously monitors a network for malicious activity and takes action to prevent it, including reporting, blocking, or dropping it, when it does occur. - Antivirus software to carry out regular scans to detect malware on staff computers
Hackers may have used SQL injection to steal the customer information from the database through the company's website	Intrusion Detection System (IDS) which takes as input the audit record of system use to identify attempted unauthorised system access or manipulation

6d Customer passwords are hashed and the hashed password is stored. 1

The password is not intended to be decrypted. The process is irreversible and is only used for user verification. 1

During subsequent user login, the user's password is hashed and verified using the hashed password stored in the database.

e **General Ethical Principles:**
Respect privacy.
Honor confidentiality. Any 1
Professional responsibilities:
Access computing and communication resources only when authorized or when compelled by public good.

8a

Static Memory Allocation	Dynamic Memory Allocation
Memory is allocated before the execution of the program begins. (During Compilation)	Memory is allocated during the execution of the program.
No memory allocation or deallocation actions are performed during Execution.	Memory Bindings are established and destroyed during the Execution. Deallocation actions managed by the programmer.
Variables remain permanently allocated.	Allocated only when program unit is active.
Implemented using stack.	Implemented using heap.
Wastage of unused space.	May have memory leak (unsuccessful deallocation).
Faster execution than Dynamic	Slower execution than static
More memory Space required.	Less Memory space required.

Advantages (any one of the following):

- Faster Access Times: Static memory allocation typically results in faster access times to user profiles. Since the memory locations of user profiles are fixed and known in advance, accessing user data can be very efficient.
- Predictable Resource Usage: The use of static memory makes it easier for the developer to predict and manage resource usage/ allocation.

Disadvantages (any one of the following):

- Memory Allocation: One of the main drawbacks is memory inefficiency. Allocating a fixed amount of memory for each user, regardless of the size of their profile, can lead to wasted memory. Users with minimal data in their profiles will occupy the same amount of memory as those with extensive profiles, potentially leading to resource underutilization.
- Scalability Challenges: As the user base of the social networking app grows, static memory allocation may become a scalability bottleneck. It can be challenging to accommodate many user profiles while maintaining efficient memory usage.

Possible impact on user experience (must be relevant to advantage and disadvantage discussed):

- Users will experience quick loading times when viewing their own or others' profiles, leading to a smoother and more responsive user interface.
- The consistency contributes to a more reliable and predictable user experience.
- If the app introduces new features or expands the types of data stored in user profiles, it may require a significant redesign of the memory allocation strategy, potentially causing disruptions to the user experience.

8b Pros:

- Memory Efficiency: Dynamic memory allocation for posts ensures that memory is only allocated when needed, reducing memory wastage.
- Scalability: The dynamic stack can grow or shrink as the number of posts fluctuates, making it suitable for accommodating many posts.

Cons

- Risk of Memory Leaks: In dynamic memory allocation, there's a risk of memory leaks if memory is not properly deallocated after a post is deleted or goes out of scope. Accumulated memory leaks can degrade app performance over time and potentially lead to crashes.

  Note:
  Memory leak is when a computer program incorrectly manages memory allocations in a way that memory which is no longer needed is not released.

- Variable Access Times: Accessing posts stored in dynamic memory can be less efficient than static memory. Since memory locations are not fixed, accessing a specific post may require more time, especially when the Timeline contains many posts. This can result in variable access times and potentially affect the user experience.

Complex Memory Management: Dynamic memory allocation for posts introduces complexity in memory management. Unlike static memory allocation, where memory is allocated once and remains fixed, dynamic allocation involves creating and deallocating memory for each post. This complexity can lead to challenges in tracking allocated memory and ensuring timely deallocation.