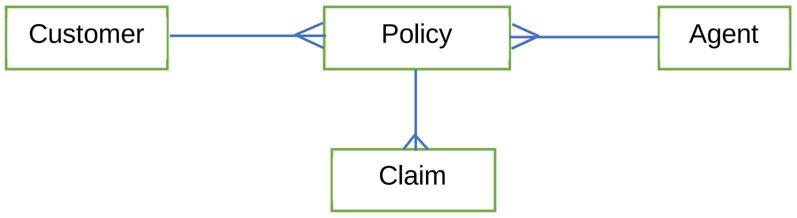
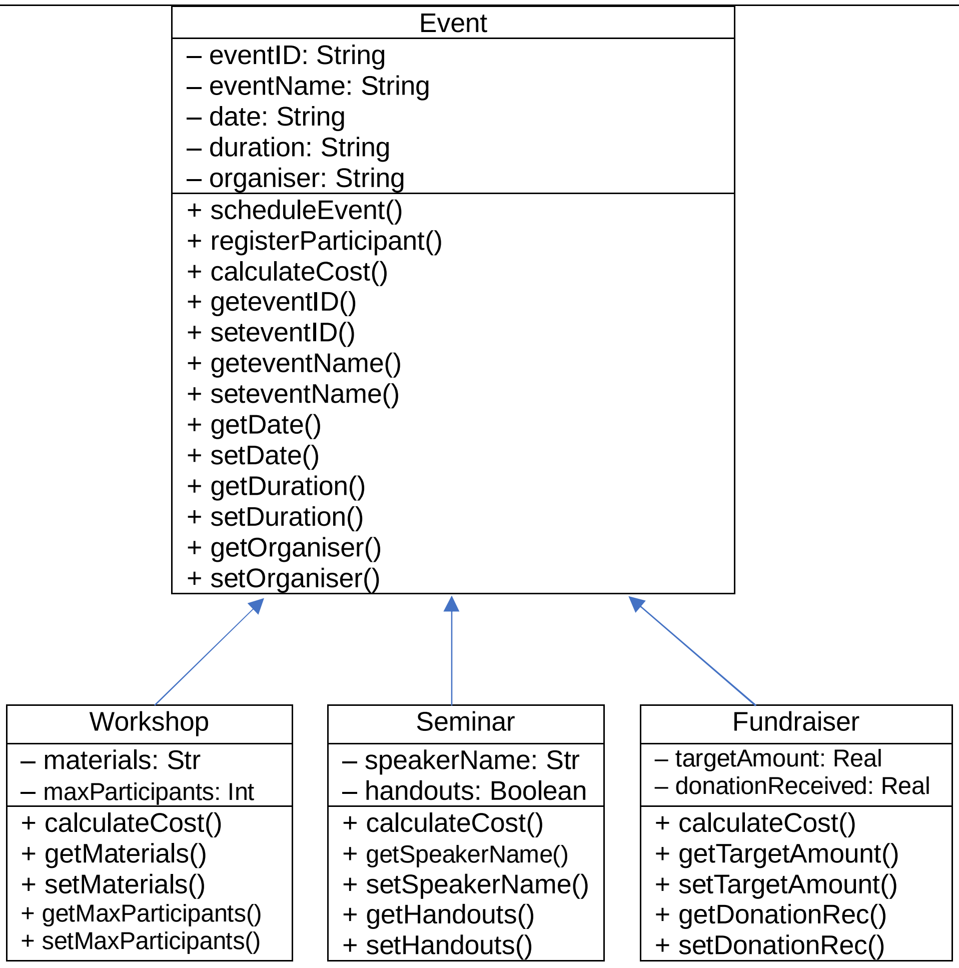
1 (a)		3 marks
(b)	Policy (**<u>PolicyNumber</u>**, PolicyType, CoverageAmount, Premium, StartDate, EndDate, *CustomerID*, *AgentID*) Customer (**<u>CustomerID</u>**, Name, ContactNumber, EmailAddress, HomeAddress) Claim (**<u>ClaimID</u>**, ClaimDate, Amount, Status, *PolicyNumber*) Agent (**<u>AgentID</u>**, Name, ContactNumber, EmailAddress)	6 marks

(c) (i)	SELECT P.PolicyType, SUM(C.Amount) AS TotalClaimAmount
	FROM Policy P
	INNER JOIN Claim C ON P.PolicyNumber = C.PolicyNumber
	GROUP BY P.PolicyType
	ORDER BY TotalClaimAmount DESC;
		4 marks

(c) (ii)	UPDATE Policy
	SET Premium = 1500
	WHERE PolicyNumber = 'P12345';
		2 marks

(d)	NoSQL databases do not require a predefined schema, allowing for the storage of unstructured or semi-structured data. Advantageous for managing diverse types of information such as customer reviews, social media interactions, and real-time claims data, which may have varying formats and structures. NoSQL databases are designed to scale horizontally by adding more servers to distribute the load. Beneficial for handling large volumes of data and high transaction rates without significant performance degradation. NoSQL databases often support hierarchical data storage, where less frequently accessed data can be moved to more cost-effective storage solutions, optimizing storage costs and performance.	3 marks
(e)	Ensure that customers are informed about and consent to their data being shared with the vendor for fraud detection purposes. Ensure data shared is encrypted and that access is restricted to authorized personnel. Notify customers about the data sharing arrangement, including how their data will be used and protected.	2 marks

2 (a)	 Event
	– eventID: String
	– eventName: String
	– date: String
	– duration: String
	– organiser: String
	+ scheduleEvent()
	+ registerParticipant()
	+ calculateCost()
	+ geteventID()
	+ seteventID()
	+ geteventName()
	+ seteventName()
	+ getDate()
	+ setDate()
	+ getDuration()
	+ setDuration()
	+ getOrganiser()
	+ setOrganiser()
	Workshop
	– materials: Str
	– maxParticipants: Int
	+ calculateCost()
	+ getMaterials()
	+ setMaterials()
	+ getMaxParticipants()
	+ setMaxParticipants()
	Seminar
	– speakerName: Str
	– handouts: Boolean
	+ calculateCost()
	+ getSpeakerName()
	+ setSpeakerName()
	+ getHandouts()
	+ setHandouts()
	Fundraiser
	– targetAmount: Real
	– donationReceived: Real
	+ calculateCost()
	+ getTargetAmount()
	+ setTargetAmount()
	+ getDonationRec()
	+ setDonationRec()
		8 marks

(b)	Enables a single method to be used in different ways depending on the object that invokes it. Demonstrated through the calculateCost() method. Each subclass inherits this method but provides its specific implementation, the way it calculates the cost varies depending on whether the event is a workshop, seminar, or fundraiser. For instance, call calculateCost() method on each subclass object, the calculateCost() method for each specific event type will be executed.	3 marks
(c)	Bundling the attributes and methods that operate on the data into a single unit, usually a class. Restrict access to the internal state of the object and only allow access through well-defined interfaces. Implemented by grouping the attributes like eventID, eventName, date, duration, organiser and methods like scheduleEvent(), registerParticipant(), calculateCost() into the Event class. The specific attributes of each event type are encapsulated within their respective subclasses. Access to these attributes is typically controlled through public methods (getters and setters), ensuring that the internal state of an object is protected from unauthorised access or modification.	2 marks

3 (a)	Array: Best Case: O(1) - If there is space available in the array, adding a new song simply involves placing it in the next available position, which takes constant time. Worst Case: O(n) - If array full, resizing is required. Involves creating a new array and copying all n elements from old array to new one. Linked List: Time Complexity: Adding a new song to a linked list (adding it to the beginning of the list in the pseudocode) has time complexity of O(1). Involves creating a new node and adjusting pointers, without need to traverse the list or resize any structure.	3 marks

(b) — 5 marks

```
FUNCTION RemoveSongFromArray(playlistArray, songToDelete,
currentSize) RETURNS playlistArray, currentSize
    foundIndex ← -1
    // Locate the index of the song to delete
    FOR i = 0 to currentSize - 1
        IF playlistArray[i] = songToDelete THEN
            foundIndex ← i
            BREAK
        ENDIF
    ENDFOR
    // If the song was found, remove it
    IF foundIndex <> -1 THEN
        // Shift elements to fill the gap
        FOR i = foundIndex to currentSize - 2
            playlistArray[i] ← playlistArray[i + 1]
        ENDFOR
        // Clear the last element (optional)
        playlistArray[currentSize - 1] ←□NULL
        currentSize ← currentSize - 1
    ENDIF

    RETURN playlistArray, currentSize
ENDFUNCTION
```

(c) — 5 marks

```
FUNCTION RemoveSongFromLinkedList(head, songToDelete) RETURNS
head
    current ← head
    previous ← NULL

    // Traverse the linked list to find the song
    WHILE current <> NULL DO
        IF current.song = songToDelete THEN
            // If the song to delete is the head
            IF previous = NULL THEN
                head ← current.next
            ELSE
                previous.next ← current.next
            ENDIF
            // Remove the node by updating the pointers
            current ← NULL
            BREAK
        ENDIF
        previous ← current
        current ← current.next
    ENDWHILE

    RETURN head
ENDFUNCTION
```

(d)	Search in an extremely large playlist. Ideal hash table O(1), compared to array O(n) and linked list O(n).	3 marks
4 (a)	The Intranet is a private, internal network accessible only to authorized employees. It hosts internal resources, such as documents, applications, and data, within the company's local network. The Intranet ensures that these resources are protected from unauthorized access by external entities. A VPN creates a secure, encrypted connection between a remote employee's device and the company's internal network (Intranet) over the public internet. The VPN masks the employee's IP address, ensuring that the communication remains private and preventing eavesdropping by malicious actors. By using a VPN to connect to the Intranet, employees working remotely can securely access internal resources as if they were physically on the company's local network. This combination provides both secure communication channels through encryption and controlled access to internal resources, ensuring that only authorized users can access the Intranet.	3 marks
(b)	Centralized Management: All data and applications are stored on a central server, making it easier to manage and update them. Efficient Use of Resources: The server handles most of the processing, allowing client devices to perform better. Consistent Access and Security: Everyone accesses the same data and applications, ensuring consistency and allowing for better security control.	3 marks
(c)	Blocking Unauthorized Access: A firewall enforces security rules to block unauthorized access attempts from external sources, allowing only approved traffic through. Common Filtering Rules: Firewalls apply rules based on packet headers (IP addresses, port numbers) and payload content (keywords, patterns) to control the flow of data and protect against malicious or unwanted traffic.	2 marks
(d)	Firewalls do not offer comprehensive protection against malware like viruses, worms, or Trojans. Additional security measures, such as antivirus software and intrusion detection systems, are required to address these threats. Firewalls can be circumvented by VPNs or other forms of encrypted traffic. VPNs encrypt all traffic, making it difficult for firewalls to inspect and filter the content. This can allow users to bypass restrictions or hide their online activities from the firewall, potentially undermining its effectiveness. Firewalls may not effectively detect or prevent insider threats, where authorized users exploit their privileges to carry out attacks or data breaches from within the network. This requires additional security measures focused on internal activity monitoring.	3 marks

5 (a)	By repeatedly dividing the problem into smaller, more manageable subproblems and then combining the solutions. Works by dividing the dataset into smaller subsets, until each subset contains only one element. The single element subset is considered sorted. The sorted subsets are merged back together in a way that maintains the sorted order.	3 marks
(b)	Compared to Insertion Sort, which has a worst-case time complexity of O(n^2), Merge Sort consistently achieves a time complexity of O(n log n). As size of the dataset increases, time taken by Merge Sort grows at a significantly slower rate, making it more efficient for large-scale sorting.	2 marks
(c)	Recursive: Simple to understand and implement; recursion naturally follows the divide-and-conquer strategy. Iterative: Avoids recursion overhead and reduces the risk of stack overflow, making it more suitable for large datasets.	2 marks

6 (a) — 4 marks

Step	node	Stack	Output
1	S	[]	
2	A	[S]	
3	null	[S, A]	
4	A	[S]	A
5	null	[S]	A
6	S	[]	A, S
7	T	[]	A, S
8	R	[T]	A, S
9	null	[T, R]	A, S
10	R	[T]	A, S, R
11	null	[T]	A, S, R
12	T	[]	A, S, R, T
13	null	[]	A, S, R, T

(b) — 4 marks

```
FUNCTION ProcessRecursive(root)
    IF root is not null THEN
        ProcessRecursive(root.left)
        print(root.value)
        ProcessRecursive(root.right)
    ENDIF
ENDFUNCTION
```

(c)	Structure: values on left subtree < root value < values on right subtree Time complexity: Searching on binary tree may be O(n) due to no order Searching on BST is O(log n) as you can narrow down the search space by half each time	2 marks
(d)	Decrease search efficiency → O(n) in worst case, like a linked list Inefficient insertions and deletions → due to increased height of tree, therefore more comparisons	2 marks

7 (a)	The loop will iterate from `0` to `NumberOfCodes`, inclusive. This causes the loop to try accessing an index that is out of bounds for the `Codes` array. To fix this issue, the loop should iterate from `0` to `NumberOfCodes - 1`, ensuring that the loop only accesses valid indices of the `Codes` array.	2 marks
(b)	Valid Codes Count: 1 (only "1234567890128" is valid) Loop through the Codes: • For i = 0 (Code: "1234567890128"): o Length: 13 (valid) o Numeric Check: All digits (valid) o Checksum Calculation: ▪ TotalSum: (1 * 1) + (2 * 3) + (3 * 1) + (4 * 3) + (5 * 1) + (6 * 3) + (7 * 1) + (8 * 3) + (9 * 1) + (0 * 3) + (1 * 1) + (2 * 3) = 92 ▪ CheckDigit: 8 ▪ Checksum Validation: (92 + 8) % 10 = 100 % 10 = 0 → Valid Invalid Codes List: ["123456789012a", "1234567890", "123456789012", "1234567890124"]	4 marks
(c)	Initialisation: Create an empty list InvalidCodes to store invalid codes. Validation Checks: Add codes to InvalidCodes if they fail length, numeric, or checksum checks. Return Value: Return a tuple containing the count of valid codes and the list of invalid codes.	3 marks

8 (a) — 4 marks

	1	2	3	4	5	6	7	8
Is it night time?	Y	Y	Y	Y	N	N	N	N
Is room occupied?	Y	Y	N	N	Y	Y	N	N
Is system manually override?	Y	N	Y	N	Y	N	Y	N
Outcome								
Lights on		X				X		
Lights off	X		X	X	X		X	X
Set to specific colour		X						

(b) — 2 marks

	1	2	3	4
Is it night time?	-	Y	-	N
Is room occupied?	-	Y	N	Y
Is system manually override?	Y	N	N	N
Outcome				
Lights on		X		X
Lights off	X		X	
Set to specific colour		X		

(c)	16	1 mark
(d)	48	1 mark
(e)	Hex 6E = Bin 0110 1110 Cooling status = ‘on’ Mode setting = ‘fan’ Temperature setting: bin 1110 (14)	2 marks
(f)	5 different rooms → need 3 bits (2^3 = 8 possible combinations), add 3 bits for the room identifier, making a total of 10 bits (7 + 3) To accommodate 10 bits, system requires at least 2 bytes (16 bits) to identify each of the 5 rooms.	2 marks