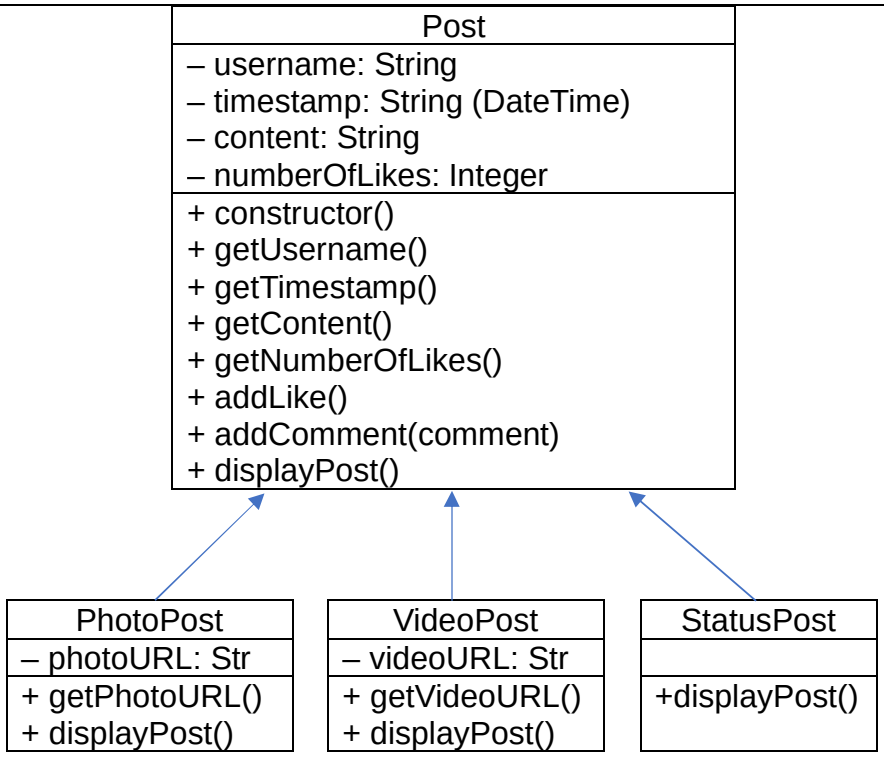
<table>
<tr><td>1
(a)</td><td> 
</td><td>3 marks
1 m for each relationship</td></tr>
<tr><td>(b)</td><td>Subject (<u>SubjectID</u>, SubjectName, Description)
Book (<u>BookNumber</u>, Title, YearPublication, SubjectID*)
BookAuthor (<u>AuthorID*</u>, <u>BookNumber*</u>, Role)
Author (<u>AuthorID</u>, FirstName, LastName, Nationality)</td><td>8 marks

2 mark each</td></tr>
<tr><td>(c)</td><td>SELECT Author.AuthorID, Author.FirstName, Author.LastName, Author.Nationality, COUNT(BookAuthor.BookNumber) AS total_written FROM Author INNER JOIN BookAuthor ON Author.AuthorID = BookAuthor.AuthorID GROUP BY Author.AuthorID ORDER BY total_written DESC;</td><td>5 mark</td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td>2
(a)</td><td> 
</td><td>8 marks</td></tr>
<tr><td>(b)</td><td>1. Each of the subclasses extends the Post base class and inherits its properties and methods. Additionally, they have their own unique attributes and methods specific to their post type.
2. Example: photoURL is property of PhotoPost to store URL of photo.
3. Allows reuse of code and extend properties and methods, make code more organised and efficient. Allows own functionality besides inheriting the common ones from base class.</td><td>3 marks</td></tr>
</table>

<table>
<tr><td>(c)</td><td>1. Add Comment class:
• Attributes: commenterUsername, timestamp, commentText
• Methods: getUsername, getTimestamp, getText
2. Modify Post class: add CommentList (array) to store comments</td><td>3 marks</td></tr>
<tr><td>(d)</td><td>• distribute the load across multiple servers or nodes
• allows app to change data structures and requirements over time
• includes diverse data types of different collections
• able to handle heavy workload quickly where users frequently access and interact with post and comments</td><td>3 marks</td></tr>
<tr><td>(e)</td><td>User consent and notification
Data minimisation and retention</td><td>2 marks</td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td>3
(a)</td><td>18</td><td>1 mark</td></tr>
<tr><td>(b)</td><td>Find the maximum value of the tree/subtree</td><td>1 mark</td></tr>
<tr><td>(c)</td><td><pre>FUNCTION FindMaximum(root) RETURNS INTEGER
    IF root = NULL THEN
        RETURN NULL
    ENDIF

    IF root.right_ptr = NULL THEN
        RETURN root.number_item
    ENDIF

    RETURN FindMaximum(root.right_ptr)
ENDFUNCTION</pre></td><td>5 marks</td></tr>
<tr><td>(d)</td><td>Frequent search operations required, choose binary search tree (BST) to store ordered data.
BST can optimize search operations – average time complexity of O(log n), efficient for frequent search operations, especially in large datasets. In contrast, searching in a singly linked list would require time complexity of O(n), which can be significantly slower for large datasets.</td><td>2 marks</td></tr>
<tr><td>(e)</td><td>Frequent insertions and deletions of nodes, a singly linked list may be preferred for storing the ordered data – adjust pointers easily. BST require rebalancing operations to maintain their structure after insertions and deletions.</td><td>2 marks</td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td>4
(a)</td><td>
<table>
<tr><td>Conditions</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td></tr>
<tr><td>At least 21 years old</td><td>Y</td><td>Y</td><td>Y</td><td>Y</td><td>N</td><td>N</td><td>N</td><td>N</td></tr>
<tr><td>Employed for at least 6 months</td><td>Y</td><td>Y</td><td>N</td><td>N</td><td>Y</td><td>Y</td><td>N</td><td>N</td></tr>
</table>
</td><td>4 marks</td></tr>
</table>

	Pass medical exam	Y	N	Y	N	Y	N	Y	N
	Actions								
	Eligible for health insurance	X							
	Eligible for probationary health insurance		X	X		X			
	Not eligible for health insurance				X		X	X	X

(b)

Conditions	1	2	3	4	5	6	7
At least 21 years old	Y	Y	Y	Y	N	N	N
Employed for at least 6 months	Y	Y	N	N	Y	Y	N
Pass medical exam	Y	N	Y	N	Y	N	-
Actions							
Eligible for health insurance	X						
Eligible for probationary health insurance		X	X		X		
Not eligible for health insurance				X		X	X

2 marks

(c)

```
FUNCTION checkEligibility(employeeAge,
monthsEmployed, passedMedicalExam):
    IF employeeAge >= 21 AND monthsEmployed >= 6
THEN
        IF passedMedicalExam THEN
            RETURN "Eligible for health insurance"
        ELSE
            RETURN "Eligible for health insurance for
a probationary period"
        ENDIF
    ELSE
        return "Not eligible for health insurance"
    ENDIF
ENDFUNCTION
```

4 marks

(d)

Backup data in case of accidental data loss due to system failures, hardware malfunctions, human errors. Company can recover most recent version of data.
Personnel records are sensitive, losing them can have serious impact on legal, economic and operational aspects.

Archiving data to store data that may not be frequently accessed for compliance to legal requirements, or for auditing purposes.

1m – explain backup
1m – explain archive
1m – explain reason

5 (a)

Function call	number	base	remainder	digit	quotient	temp String	Return value
1	42	16	**10**	**A**	**2**		

4 marks

	2	**2**	**16**	**2**	**2**	**0**			
	3	**0**						**""**	
	2						**""**	**2**	
	1						**2**	**2A**	

(b)	Converts a decimal number into its representation in a specified base	1 mark
(c)	2 to 16	2 marks Must be range
(d)	Non-Negative Number Check: greater than or equal to zero. Prevent the conversion of negative numbers, which may not be meaningful or valid in the context of certain base conversions. Integer Input Check: has no fractional part or decimal portion or non-integer. Base conversions are applied to only whole numbers, not non-integers. Range check: 2 to 16. Ensure that the range of base values are valid.	2 marks
(e)	By using the call stack to keep track of each recursive call. For each recursive function call, a new frame is pushed onto the call stack To store local variables and the return address (point in the code where the function needs to resume) when the recursive call returns. When a base case is reached (in this case, when the number becomes zero), the function starts returning values and popping frames off the call stack. This process continues until the initial function call returns, at which point the call stack is empty.	3 marks
6 (a)	• browser first checks its local cache to see if it already knows the corresponding IP address for that URL. • If not, it queries a Domain Name System (DNS) server. • The DNS server maintains database that maps domain names to IP addresses. • It looks up the IP address associated with the provided URL and returns it to the browser. • IP address is then used to establish a connection to the web server.	3 marks
(b)	Web server needs to be easily accessible on the internet. A fixed known address for the server make it straightforward for users to find and connect to it consistently. Client computers are often assigned dynamic IP addresses by DHCP (Dynamic Host Configuration Protocol) server, allowing efficient utilisation of available IP addresses within a network.	2 marks

(c)	• Parse request to extract HTTP method, headers, resource path (e.g., a web page or an application). • Route request to appropriate handler based on resource path • Executes any server-side scripts based on requested resource • Retrieves requested resource or generates data dynamically if needed. • Sends resource back to the client over the internet using the HTTP protocol. Typically includes the requested content, HTTP headers, and a status code indicating the success or failure of the request.	4 marks
(d)	Advantages: • Highly scalable, allowing additional clients to connect to the server easily. Suitable for accommodating a large number of users or devices. • Server-side management facilitates centralised control over data, security, and updates. Easier to maintain and secure the system. Disadvantages: • If server experiences downtime or becomes inaccessible, client functionality can be severely impacted. • Implementing and maintaining a client-server architecture can be complex and require robust infrastructure, require more manpower and money.	4 marks
(e)	Developed for specific operating system of the machine or device. Access via browser and function according to machine or device. Downloaded and installed on machine. No need to be downloaded or installed. May work offline. Need an internet connection to run. Run relatively faster since data resides locally. Need to access data from server, hence relatively slower.	2 marks
(f)	Suitable for users with limited access to internet connection. Offer a more consistent user interface that is not dependent on a web browser. More personalized content and recommendation based on user preferences.	2 marks

7 (a)	• Both break down a complex problem (sorting a list) into smaller sub-problems, • solve those sub-problems independently, • and then combine the results to obtain the final sorted list. In merge sort, achieved through recursive splitting and merging. Split into two equal halves, until base case, where each subarray consists of only 1 element. Then subarrays are merged together into a single sorted array. In quick sort uses recursive partitioning and sorting. Selects a pivot element and partition array into 2 subarrays – elements less than pivot and elements greater than pivot. Partition continues for each subarray. Sorting process involves selecting new pivots and partitioning the subarrays until all elements are sorted. Then combined together where entire array becomes sorted.	4 marks
(b)	Consistently divides the data into equal halves, regardless of the initial order. Each division results in two subproblems of roughly the same size. Consequently, the time complexity remains O(n log n), where n is the number of elements in the dataset.	2 marks
(c)	Pivot chosen for partitioning consistently results in highly unbalanced partitions. This occurs when the pivot is either the smallest or largest element in the list in every partitioning step. When data is already sorted in ascending or descending order, and pivot selection consistently chooses the smallest or largest element as pivot. This occurs because in each partitioning step, it separates the data into one subarray of size n-1 and another subarray of size 1.	3 marks
(d)	Guaranteed to performed in O(n log n) time regardless of initial order of the data. Much faster than quick sort which has a worst-case time complexity of $O(n^2)$.	1 mark
(e)	An in-place sorting algorithm, requires less additional memory for temporary storage compared to merge sort. Makes quick sort more memory-efficient for sorting large datasets, especially in situations where memory resources are limited.	1 mark