	2022 JC 2 H2 Computing Mock Practice 2 (Paper 1)
1	**(a)** Front – stores an integer that represents the array index of the first element in the queue. Rear – stores an integer that represents the array index of the last element in the queue. **(bi)** `Size` and `Limit` are used to store integer values that will keep track if the queue is empty, full, or initialised. Assuming queue has some items but not full since an element can be added into it: `Rear = 1, Front = 0, N = 2, Size = 2` To **<u>add</u>** an item to queue: • Check if queue is full `size >= limit` • `Rear = Rear + 1` • `Size = Size + 1` • `Store item into index queue[Rear]` **(bii)** `Size=0`, `Limit=N`, `Front=0`, and `Rear=-1` Assuming queue has some items but not empty since an element can be removed from it To **<u>remove</u>** an item from queue: • Check if queue is empty `Size = 0` • `Return item queue[Front]` • `Front = Front + 1` • `Size = Size - 1`
	(c) Each add/remove operation will increment `Rear` or `Front` by 1 respectively and the values will reach `Limit` eventually. Which will result in available spaces not utilised.
	(c) **<u>To add:</u>** Check queue not full (`size < limit`) `Rear = (Rear + 1) MOD Limit`

	Store item into `queue[rear].` Increment `Size` **To delete:** Check queue not empty (`size > 0`) Return `queue[Front]` `Front = (Front + 1) MOD Limit` Decrement `Size`
	(e) • Printer spool • OS job processing queue • Simulation software
2	**(a)** Encapsulation – Is the combination of attributes and methods as a single object type. **(b)** Inheritance – Is the ability for one of the class objects to acquire attributes or methods from the parent class or higher. **(c)** Data hiding – Having processing of data within the object itself privately. This allows internal implementation of the object to be change only to the application that uses it. Accessing or modifying to attributes of the class can only be done via the class's assessor methods. **(d)** Polymorphism – The ability of parent classes and classes to respond to the same method in different ways.
3	**(a)** Recursion is a way of programming to solve a problem where a • subroutine is able to call itself one or more times in its body (general case), • and then terminates when it reaches a stopping condition (base case). Iteration is another way of programming that uses loops to repeat its processes
	(b) Binary search The data set has to be sorted into a pre-defined order.
	(ci) `'Handel','Prokofiev', 'Vivaldi'` **(cii)**

	`'Handel', 'Bartok', 'Bach'`
	(di) $O(\log_2 N)$
	(dii) $\log_2 678 = \log_{10}(678) / \log_{10}(2)$ $= 9.41$ Therefore maximum number of comparisons is 10
4	**(a)** 1. The ordering process will classify the sequence into a sorted list and an unsorted list. 2. It begins by placing the first item into the sorted list as already sorted (eg. leftmost item) Sorted: 35 Unsorted: 57, 19, 59, 48, 89, 30 3. Subsequently for every iteration, **each element in the unsorted part will be compared sequentially item by item with every element in the sorted part**, until the correct **position for the unsorted element is found for insertion**. The **iteration continues until** the whole list is sorted or **unsorted part becomes empty**. Sorted: 35, 57 Unsorted: 19, 59, 48, 89, 30 Sorted: 19, 35, 57 Unsorted: 59, 48, 89, 30 Sorted: 19, 35, 57, 59 Unsorted: 48, 89, 30 Sorted: 19, 35, 48, 57 Unsorted: 89, 30 Sorted: 19, 35, 48, 57, 89 Unsorted: 30 Sorted: 19, 30, 35, 48, 57 Unsorted: Null
	(b)

	As time complexity of insertion sort generally is $O(n^2)$, one should consider insertion sort when the size of the data set is small or the data set is almost pre-sorted.
	(c) For worst case, the initial data set comes as a sequence sorted in the reversed order. There in an array of N items: 1st iteration: 1 comparison (1 item already in sorted list) 2nd iteration: 2 comparisons 3rd iteration: 3 comparisons Nth iteration: N-1 comparisons Total no of comparisons = 1+ 2+3+…+N-1 = (n/2)(a+N–1) → $O(n^2)$
5	(a) 1. Data Accuracy - relational database uses primary keys and foreign keys to make the tables interrelated to each other. Thus, all the data which is stored is **non-repetitive**. Therefore, the data stored can be guaranteed to be accurate. 2. Data independence – data is stored in a way that changes to the structure of the database will not affect any of the programs that access the data. 3. Data consistency – each individual data item is only held at one place as a record. Any updates will only be done once and there will not be a fear of updating performed at one system and not another. 4. Data Security – A DBMS provides a framework for better enforcement of data privacy and security policies. Offers user authentication and allows different levels of access for different users. 5. Multiple users access (improved data sharing) - creates an environment in which end users have better access to more and better-managed data. Ensures that multiple users can connect to the database simultaneously, and enables the users to manipulate/ update/ alter the data all at the same time without causing the database to crash. 6. Backup & recovery - Most relational databases offer easy export and import options, making backup and restore trivial and simple.
	(b) NRIC number is permanent and irreplaceable, it is generally considered as more **sensitive personal data** and therefore requires a higher level of protection.

	Improper handling of an individual's NRIC or NRIC number increases the risks of such data being used for illegal purposes such as fraud or identity theft. Solution: Do not collect NRIC as it is not the only key used to recognise a student uniquely. Instead, consider a composite key made up by selecting a part or the entire attribute concatenated with last 4 characters of NRIC.
	(c) • Primary key uniquely identifies each record in a table. • Primary keys must contain UNIQUE values, and cannot contain NULL values. • A table can have only ONE primary key; and in the table, this primary key can consist of single or multiple columns (fields).
	(d) (Normalisation is a technique of organising the tables in a database to reduce data redundancy and prevent inconsistent data. There are at least three normal forms associated with normalisation: first normal form (1NF), second normal form (2NF), and third normal form (3NF).
	(e) 
	(f) CLASS(Class ID, Civics Tutor) STUDENT (Ident_No, StudentName) LOAN_BY_STUDENT (LoanID, DateOfBorrow, DateOfRet, Due_Date, Ident_No(fk)) LOANED_Device(Loan_ID(fk), SerialNo (fk)) Library Item (SerialNo, Type, Brand, Model, Remarks)
	(g) Data validation checks if the data matches in context to the specifications. Eg. checks for ClassID to be in range of 1 to 10. Data verification is a process in which different types of data are checked for accuracy and inconsistencies after data migration is done. It helps to determine whether data was

	accurately translated when data is transferred from one source to another, is complete, and supports processes in the new system. Eg double entry
6	(a) Date of vehicle entry Time of vehicle entry
	(b) A hash function uses the record key to calculate the suggested index address of a record for data storage or retrieval.
	(c) `hash(1125795021)= 1125795021 MOD 563 = 516` Inspect if `hashtable[516]` is empty. Since `hashtable[516]` is empty, store vehicle record with IU number 1125795021 into `hashtable[516]`
	(d) Collisions happens when the same hash function hashes 2 or more distinct vehicle entry records to the same location in the hash table.
	(e) In linear probing, • If a key creates a hash value that references a position that is already occupied, then the data will be referenced sequentially, probing to the **next position** until an empty slot is found in the hash table.
	(f) Linear probing has a tendency to create long runs of occupied slots from the hash position, referred to as a cluster. The bigger the cluster gets, the more likely it grows, thus reduces the performance of operations on the hash table.
7	(a) • A binary search tree is a binary tree where each node contains a value from a well-ordered set. • Every node in the left subtree of n contains a value which is smaller than the value in the node n, and every node in the right subtree of n contains a value which is larger than the value in the node n.

<table>
<tr><td></td><td>(b)
X = 0</td></tr>
<tr><td></td><td>(c)
<table>
<tr><th>Location</th><th>Left</th><th>Data</th><th>Right</th></tr>
<tr><td>6</td><td>0</td><td>Colin</td><td>0</td></tr>
<tr><td>5</td><td>0</td><td>Clarence</td><td>6</td></tr>
<tr><td>4</td><td>0</td><td>Albert</td><td>5</td></tr>
<tr><td>3</td><td>0</td><td>Vincent</td><td>0</td></tr>
<tr><td>2</td><td>0</td><td>Ken</td><td>3</td></tr>
<tr><td>1</td><td>4</td><td>Nigel</td><td>2</td></tr>
</table>
Head = 1
Free = 0</td></tr>
<tr><td>8</td><td>(ai)
6,8,9,11,12,14,15,20,26,30,35</td></tr>
<tr><td></td><td>(aii)
15,11,8,6,9,12,14,26,20,30,35</td></tr>
<tr><td></td><td>(aiii)
6,9,8,14,12,11,20,35,30,26,15</td></tr>
<tr><td></td><td>(b)
895-/24*+</td></tr>
<tr><td>9</td><td>(a)

In a client-sever network, there exist a centralised and high performing computer, known as the server that dedicatedly serves as a hub to many other computing devices (eg. PCs, printers, fax, etc), known as the clients.
A connection between a client and the server will be established when a client requests to communicate with the server. Once the server has fulfilled a client's request, the connection will be terminated.
</td></tr>
<tr><td></td><td>(b)

File server

Server that manages the storage and access to files.
Central file storage location that can be accessed by multiple clients/networks (eg. School network, SME network, home network).

Web server

Hosts websites.
Responds to clients’ requests on the website it hosts.

</td></tr>
</table>

	○ Dedicated web servers are more reliable as they host lesser websites, reducing the occurrences of issues like bottleneck