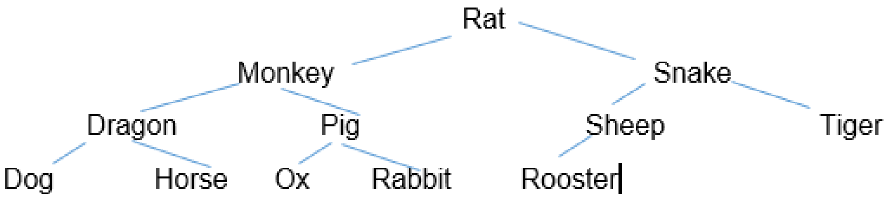
2021 YIJC H2 Computing J2 Prelim Exam Paper 1 (Theory)

Suggested Solutions and Marking Scheme

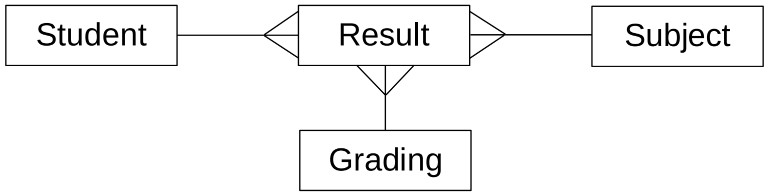
<table>
<tr><th colspan="2">Qn</th><th>Solution</th><th>Marking Scheme</th></tr>
<tr><td><b>1</b></td><td><b>(a)</b></td><td>
</td><td>[1] Dragon Subtree
[1] Monkey + Pig subtree
[1] Rat + Snake subtree</td></tr>
<tr><td></td><td><b>(b)</b></td><td>(i) Function call: <code>Fn(X,'Sheep')</code>
<table>
<tr><th>Curr</th><th>Object = Arr[Current]</th><th>Object > Arr[Current]</th><th>Next</th></tr>
<tr><td>1</td><td>Sheep=Rat; F</td><td>Sheep&gt;Rat; T</td><td>3</td></tr>
<tr><td>3</td><td>Sheep=Snake; F</td><td>Sheep&gt;Snake; F</td><td>6</td></tr>
<tr><td>6</td><td>Sheep=Sheep ; T</td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td colspan="4">Output : 6</td></tr>
</table>
(ii) Function call: <code>Fn(X,'Duck')</code>
<table>
<tr><th>Curr</th><th>Object = Arr[Current]</th><th>Object > Arr[Current]</th><th>Next</th></tr>
<tr><td>1</td><td>Duck=Rat; F</td><td>Duck&gt;Rat; F</td><td>2</td></tr>
<tr><td>2</td><td>Duck=Monkey; F</td><td>Duck&gt;Monkey; F</td><td>4</td></tr>
<tr><td>4</td><td>Duck=Dragon; F</td><td>Duck&gt;Dragon; T</td><td>9</td></tr>
<tr><td>9</td><td>Duck=Horse; F</td><td>Duck&gt;Horse; F</td><td>18</td></tr>
<tr><td colspan="4">Output : -1</td></tr>
</table></td><td>[1] Boolean
[1] Next: 3&6
(No marks deducted for filling up row 3)
[1] Boolean
[1] Next (Award for ECF)
[1] output]</td></tr>
<tr><td></td><td><b>(c)</b></td><td>It attempts to search for a string in the array arr doing a binary search, traversing a BST
and returns it’s index, otherwise it returns -1 if not found</td><td>[1] Binary search
[1]</td></tr>
<tr><td></td><td><b>(d)</b></td><td>Linear search requires visiting all the elements sequentially until the search key is found (or O(n))
Fn uses binary search to reduce the problem (array size) by half after each comparison (or O(lg n)).</td><td>[1]
[1]</td></tr>
</table>

	(e)	Array Y Y[1] 'Rabbit' Y[7] 'Tiger' Y[2] 'Horse' Y[8] 'Dog' Y[3] 'Snake' Y[9] 'Goat' Y[4] 'Dragon' Y[10] 'Monkey' Y[5] 'Ox' Y[11] 'Pig' Y[6] 'Rooster' Y[12] 'Rat'	[1] At least 9 correct [1] All correct
2	**(a)**	It is recursive because Lines 2 to 9 contains code for the base case and lines 13 and 15 call the function again recursively reducing it to a smaller sub-problem.	[1] [1]
	(b)	It is unstable. For elements of equal value, they will be swapped in line 11 and 15. Relative position of elements of equal values have changed after the sort. The first element in the right half will be placed in front of the element in the left half, for e.g. Merging [4,5^1,7][3,5^2,8] will lead to [3,4,5^2,5^1,7,8] (example)	[1] [1]

	(c)	`FUNCTION merge_sort(LIST: seq) RETURNS LIST`
		`   IF LENGTH(seq) < 2`
		`     THEN`
		`       RETURN seq`
		`     ELSE`
		`       mid <- LENGTH(seq) DIV 2`
		`       left <- merge_sort(seq[:mid])`
		`       right <- merge_sort(seq[mid:])`
		`       RETURN merge(left, right)`
		`   ENDIF`
		`ENDFUNCTION`
			[1] Base Case
			[1] mid
			[1] recursion
			[1] return merge

	(d)	Merge sort has a faster time complexity O(n lg n) vs O(n^2) for bubble sort. Merge sort is not in-place and requires additional memory space to sort. Optimised bubble sort has a better best case time complexity of O(n) vs O(n lg n) for merge sort. Merge sort is a more complex algorithm to write code as compared to bubble sort.	[1] [1] [rejected answers, too insignificant or trivial]
	(e)	Bubble sort is more efficient because the elements are nearly in order, with some out of place elements that just require 1 swap to be corrected. Optimised bubble sort will use 2 passes to sort it correctly, so the time complexity is O(n), much better than merge sort which remains at O(n lg n).	[1] [1] O(n) or 2 passes [1]

Qn		**Solution**	**Marking Scheme**
3	**(a)**	ADT consists of a set of data and operations that can be performed on the data. Users are not concerned about how the task is done but rather with what it can do.	[1] [1]
	(b)	Advantage: Any one of the followings: - Unlimited size - Easy to delete the first (or top) element in the linked list Disadvantage: Any one of the followings: - Need to use pointers, need more memory - Difficult to traverse a linked list. .	[1] [1]
	(c)	Advantage: Any one of the followings: - Simple to implement an array - Can access an element in an array easily using the index Disadvantage: Any one of the followings: - Limited size - Difficult to remove the first (or top) element from the array	[1] [1]
	(d)	1. create a new node for the data to be inserted 2. set the next pointer to the current node at the top 3. update the top pointer to point at the new node	[1] [1] [1]
	(e)	1. pop all the elements from the original stack and push in to another stack 2. use a counter to count the number of element 3. once the original stack is empty, pop from the other stack and push all the elements back into the original stack	[1] [1] [1]
	f(i)	Elements are added into the stack such that they are sorted in ascending order. When inserting an element, all the smaller elements are recursively pop out of the stack first then insert the element; after which, the smaller elements are added back into the stack.	[1] [1]
	f(ii)	Line 04: `if stk==[] or item < stk.peek():` Line 05: `stk.push(item)` Line 07: `temp = stk.pop()` Line 09: `stk.push(temp)` or Line 04: `if stk==[] or item < peek(stk):` Line 05: `push(stk,item)` Line 07: `temp = pop(stk)` Line 09: `push(stk,temp)`	[1] [1] [1] [1]

Qn		Solution	Marking Scheme
4	(a)		[1] 4 correct entities [1] Student-Result [1] Subject-Result [1] Grading-Result

	(b)	Student(NRIC, Name, CG, Index)
		Subject(Code, Name)
		Result(NRIC, Code, Mark)
		Grading(Mark, Grade)
			[1] with PK
			[1] with PK
			[1] with PK
			[1] Mark as FK
			[1] with PK

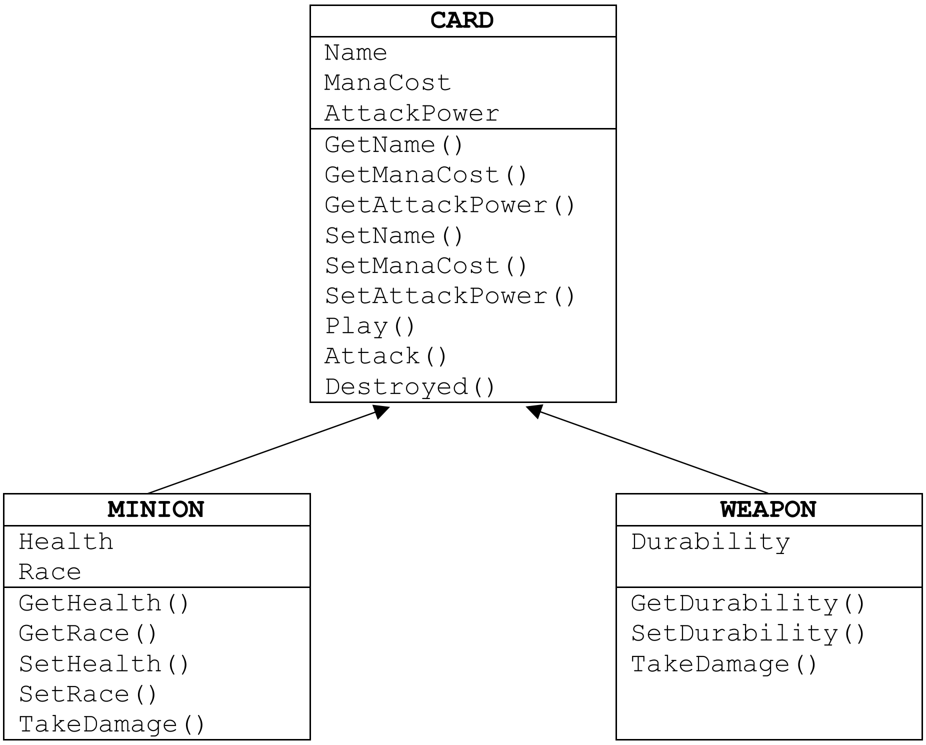
	(c)	- AdmissionID (autoincrement integer which is system generated when a student is admitted to the school) - StudentID (e.g. first 4 letters of the student's name, followed by the student's index number) Using either one of the above is less individual identifying as it is not used in any national classified/restricted government/citizen services.	[1] suitable alternative [1] logical justification

	(d)	SELECT Student.Name, CG, Grade
		FROM Student, Subject, Result, Grading
		WHERE Result.Code='9569' AND Result.Mark>=60
		AND Result.NRIC=Student.NRIC
		AND Result.Code=Subject.Code
		AND Result.Mark=Grading.Mark
			[1] correct output
			[1] from all tables
			[1] correct condition
			[1] correct links for all the tables

Qn		Solution	Marking Scheme
5	**(a)**	Client-server describe a communication between the client's program requesting a service or resource from the server's program.	[1]
	(b)	- web service with the server program code - database for menu, orders and payment	[1] [1]
	(c)	Customer uses a **browser interface [1]**, to browse the **ordering form [1]**, submit the customer's order by the **tablet's input devices [1]**, eg touch screen or with a pointing device * merely describing the whole ordering process get no mark.	[1] for each keyword or equivalent. Total [3]
	(d)	Any 1 of the following or any reasonable answer: 1. scrolling up and down -> mental model 2. a tag for each category of food item -> mental model 3. flipping thru the pages -> mental model 4. radio button or checkbox -> affordance of selection 5. use of icons like 'Chef Recommendation' -> visual hierarchy 6. with pictures showing the food items -> simplicity and clarity 7. use of different sizes of headers -> visual hierarchy	[1] [1] state the usability principle
	(e)	Any 2 of the followings or any reasonable answer: 1. IT replacing jobs: can require less people/reduce headcount of waiters, waitresses, cashiers, <u>putting them out of work</u>. 2. <u>Loss of inclusivity</u>: There may be the elderly or uneducated people among the customers who may be unable to adapt to the new method of ordering. 3. Those without online banking facility or credit card may have <u>problem with payment</u>.	[1] each Total [2]
	(f)	Any one of the followings: Native application is able to utilise the functions on the customer's personal device. This would remove the need for the customer to scan the QR code to make payment, the customer could easily use their online banking to make payment. Touchscreen (Invalid answers: GPS, camera, Bluetooth ...) - to capture thumbprint or signature for online payment. Save the previous orders in their personal devices - for easy reference, especially for returning customer. Customer able to log in with their credential - to allow ordering and transacting using their identity .	 [1] [1]

	(g)	Any one of the following: - Customer can order from the website using a browser without the need to download or install any software. - Customer can order remotely without being in the restaurant	[1]
	(h)	Diagram should include the following: - switch [1] connecting the web server, manager's computer, computer station for service crew and kitchen large display [1] - printer to print receipts - wifi access point [1] for customers' tablet device To accept online orders: router [1] and modem [1] connecting to ISP for internet service	[5] No mark awarded for connecting the device wrongly.
	(i)	Any 2 of the following benefits: - Less manpower required to hire (eg cashier, service crew) - Digitised records, ease of analysis, processing, accounting. - Going green, no need to print menu, - Going cashless, save the trouble of depositing to the bank - broaden customer base if accept online ordering	[1] each Total [2]
	(j)	Any 2 of the following PDPA Obligations: - Consent Obligation: the data collected must be used only for the purpose consented by the customer, ie for ordering and delivering of the food. - Protection Obligation: the restaurant must protect the data to prevent any unauthorised access to it; physical copies of the data should be locked away. - Retention Obligation: once the data is no longer required, eg the food has been delivered and the all payments have been completed, then the restaurant should remove the data from their system. - Transfer Limitation: the restaurant must obtain consent from the customer before disclosing the data to some third parties. - Accountability Obligation: making information about your data protection policies, practices and complaints process available upon request and designating a data protection officer (DPO) and making the business contact information available to the public.	First obligation: [1] State [1] Describe Second obligation: [1] State [1] Describe Total [4]

Qn		Solution	Marking Scheme

6	(a)	 CARD
		Name
		ManaCost
		AttackPower
		GetName()
		GetManaCost()
		GetAttackPower()
		SetName()
		SetManaCost()
		SetAttackPower()
		Play()
		Attack()
		Destroyed()
		MINION
		Health
		Race
		GetHealth()
		GetRace()
		SetHealth()
		SetRace()
		TakeDamage()
		WEAPON
		Durability
		GetDurability()
		SetDurability()
		TakeDamage()
			[1] CARD as base class + MINION or WEAPON as sub-classes
			[1] upward pointing arrow for inheritance
			[1] properties for base class CARD
			[1] properties for both sub-classes
			[1] at least one pair of getter/setter from the base class
			[1] at least one pair of getter/setter from the sub-classes
			[1] any other appropriate methods

	(b) (i)	Encapsulation is the bundling of both properties (e.g. `Name`, `ManaCost`, and `AttackPower`) and the methods (e.g. `GetName()`, `SetName()`, …) within one unit in the class `CARD` **OR** Encapsulation is the restriction of direct access to an object's private properties (e.g. `Name`, `ManaCost`, and `AttackPower`) such that they can only be accessed by public methods (e.g. `GetName()`, `SetName()`, …)	[1] definition [1] example from one of classes
	(b) (ii)	Inheritance is the ability to derive new classes by inheriting the data and operations of existing classes. The shared knowledge of common traits forms the basis for inheritance relationships, usually hierarchical from general to more specific classes. For instance, class `MINION` inherits all its properties and methods from `CARD`. In addition, it has a specific property of `Race` that some `CARD` objects may not have.	[1] definition [1] example of a sub-class in relation to a base class
	(b) (iii)	Polymorphism is the ability of methods with the same name to behave according to their own context. For example, the `TakeDamage` method for the class `MINION` will behave differently from the same name `TakeDamage` method in the class `WEAPON`. A minion takes damage when it is being attacked, while a weapon takes damage when it is being used for an attack.	[1] definition [1] example which shows appropriate polymorphic method
	(c)	OOP provides the following benefits: • more realistic model of entities as embodiment of state and behaviour • better protection of data via encapsulation • code reuse via inheritance • code generality via polymorphism	[1,1] any two benefits of OOP