National Qualifications 2019

2019 Computing Science

Advanced Higher

Finalised Marking Instructions

These marking instructions have been prepared by examination teams for use by SQA appointed markers when marking external course assessments.

General marking principles for Advanced Higher Computing Science

This information is provided to help you understand the general principles you must apply when marking candidate responses to questions in this paper. These principles must be read in conjunction with the detailed marking instructions, which identify the key features required in candidate responses.

(a) Marks for each candidate response must **always** be assigned in line with these general marking principles and the detailed marking instructions for this assessment.

(b) Marking should always be positive. This means that, for each candidate response, marks are accumulated for the demonstration of relevant skills, knowledge and understanding: they are not deducted from a maximum on the basis of errors or omissions.

(c) If a specific candidate response does not seem to be covered by either the principles or detailed marking instructions, and you are uncertain how to assess it, you must seek guidance from your Team Leader.

(d) Marks should be awarded regardless of spelling as long as the meaning is unambiguous.

(e) Candidates may answer programming questions in any appropriate programming language or pseudocode. Marks should be awarded, regardless of minor syntax errors, as long as the intention of the coding is clear.

(f) Where a question asks the candidate to **describe**, the candidate must provide a statement or structure of characteristics and/or features. This should be more than an outline or a list. It may refer to, for instance, a concept, process, experiment, situation or facts in the context of, and appropriate to, the question. The candidates will normally be required to make the same number of factual/appropriate points as there are marks available for the question.

(g) Where a question asks the candidate to **explain**, marks should only be awarded where the candidate goes beyond a description, for example by giving a reason, or relating cause to effect, or providing a relationship between two aspects. These will be related to the context of the question or a specific area within a question.

(h) Credit should be given where a labelled diagram conveys clearly and correctly the response required by the question.

Marking instructions for each question

Question			Expected response	Max mark	Additional guidance
1.	(a)		`CLASS WorkMeeting INHERITS Event WITH { STRING meetingTitle, REAL mileage, REAL travelExpenses }`	**2**	1 mark for class declaration indicating use of inheritance 1 mark for additional attributes; only award mark for nine attributes if no inheritance indicated
	(b)		The constructor assigns initial values upon instantiation of an object.	**1**	Description must mention creation/instantiation of an object.
	(c)	(i)	A new object of the `WorkMeeting` subclass has been instantiated **(1 mark)** The values are assigned to the relevant instance variables with the constructor initialising an empty array to store the list of participants and initialising the array index as zero **(1 mark)**	**2**	Award 1 mark each for 2 of the following: • instantiation/creation of object or invoke constructor method for WorkMeeting class • assignment using the 7 values provided and additional two attributes of Event class
		(ii)	Award 1 mark each for any two of the following: • A new participant called Chao Li has been added to the array of meeting participants • and the array index is incremented • Invoke addParticipants() method	**2**	
		(iii)	Encapsulation means that a property cannot be edited directly. **(1 mark)** A method needs to exist in order to update the venue property. **(1 mark)**	**2**	1 mark for encapsulation/private variable 1 mark for need for method

Question			Expected response	Max mark	Additional guidance
	(d)	(i)	`SET THIS.travelExpenses TO THIS.mileage * 0.5`	**1**	1 mark for correct calculation Note: use of THIS is not mandatory

		(ii)	`FOR i FROM 0 TO 99 DO`
			`IF (myMeetings2019[i].getDate() >= "01/04/2019") AND (myMeetings2019[i].getDate() <= "30/04/2019") THEN`
			`SET THIS.expenses TO THIS.expenses + myMeetings2019[i].getTravelExpenses()`
			`END IF`
			`END FOR`
				3	1 mark for correct use of `getDate()` and `getTravelExpenses()` methods of the array of objects
					
					1 mark for correct condition
					
					1 mark for updating running total using `myMeetings2019` array
					
					Note: accept `calculateTravelExpenses()` method rather than `getTravelExpenses()` method
					
					Note: alternative methods for finding dates in April are acceptable eg use of substring

	(e)		Increase in energy used to process users' data in the company data centre leading to increased carbon emissions. Need to use low carbon or energy efficient equipment in the company data centre to reduce carbon emissions needed to run the devices.	**2**	1 mark each Note: answers must refer to environmental impact.

Question			Expected response	Max mark	Additional guidance
2.	(a)		`<FORM action="form1.php" method="post">`	**2**	1 mark for `action` 1 mark for `method`

	(b)		`CREATE TABLE Tour (tourID int AUTO_INCREMENT PRIMARY KEY;`
			`fullName varchar(20) NOT NULL;`
			`contactNo varchar(11) NOT NULL;`
			`tripDate date NOT NULL;`
			`tripTime time NOT NULL;`
			`guideID int;`
			`FOREIGN KEY (guideID) REFERENCES Guide(guideID) );`
				3	1 mark for primary key with auto increment and correct data type
					
					1 mark for additional 5 fields with correct data types
					
					1 mark for foreign key `guideID`

	(c)	(i)	`SELECT guideID AS [Guide ID], COUNT(*) AS [Number of Tours] FROM Tour WHERE tripDate = "16/04/2019" GROUP BY guideID;`	**2**	1 mark for correct fields with `WHERE` clause 1 mark for correct use of `COUNT` and correct grouping Note: field names should match those introduced in the `CREATE` statement in part (b) Note: use of aliases is not mandatory
		(ii)	`SELECT MAX([Number of Tours]) FROM ci;`	**2**	1 mark for use of `MAX` 1 mark for correct use made of (c)(i) query

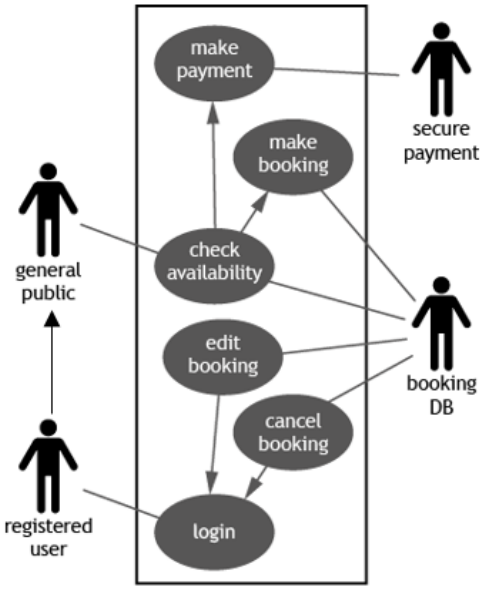
Question			Expected response	Max mark	Additional guidance
	(d)	(i)	6.1 `$servername="ice1";` 6.2 `$username="u1";` 6.3 `$password="p1";` 6.4 `$database="facilityTour";` 7.1 `$conn = mysqli_connect ($servername, $username, $password, $database);`	**2**	1 mark for assignment of data values to connection variables 1 mark for connection code Note: accept mysqli_connect without $conn variable

		(ii)	`$sql = "INSERT INTO Tour (fullName, contactNo, tripDate, tripTime, guideID) VALUES ($fullname, $contact, $date, $time, $guideID)";`
			
			`mysqli_query($conn, $sql);`
				3	1 mark for `INSERT` query with values in same sequence as fields
					
					1 mark for correct fields (`tourID` should not be listed)
					
					1 mark for execution of query

	(e)		For example: • Need to gain the necessary agreement/consent of customers whose data is gathered – without this the company is in breach of GDPR • The company must own IPR all of media used in promotional materials – without this the company is in breach of Copyright, Design and Patents Act	**1**	1 mark for any valid concern

Question			Expected response	Max mark	Additional guidance
3.	(a)	(i)	Having identified that towel B432 is out of place, the insertion sort algorithm: • works backwards down the array to identify the correct insertion position • towel B432 is copied and the towels from the insertion point upwards move one position to the right • towel B432 is then inserted into the correct position in the array Alternatively • compares towel B432 with towel F411 and swaps them • compares towel B432 with towel F127 and swaps them • compares towel B432 with towel B224 and doesn't swap then because they are in the correct sequence and towel B432 is now in the correct position in the array	**3**	1 mark each for the following: • indication of working backwards from towel B432 to the insertion point • terminating the backwards comparisons once towel B432 has been positioned correctly 1 mark for <u>either</u> of the following: • swapping towel B432 with previous elements of the array • copying towel B432, moving elements with larger values to the right and then inserting towel B432 in the correct position Note: marks can be allocated to relevant lines of responses that are presented as an algorithm; to receive full marks, the algorithm <u>must</u> refer to the scenario (eg make use of towel B432)
		(ii)	7.1 if middle element of towelID array = towelIDtoFind then 7.2 set found to true 7.3 else if middle element of towelID array < towelIDtoFind then 7.4 set startPointer TO middle+1 7.5 else 7.6 set endPointer TO middle-1 7.7 end if	**3**	1 mark for each correct comparison and output Note: middle must be used as an array index and not a comparatpr Note: 3 separate IF statements are also acceptable
		(iii) A	Recursion	**1**	
		(iii) B	The values of the first 3 elements in the quantity array are totalled. The array index is reduced each time the function is called until the base case is reached.	**2**	1 mark for totalling the quantity of towels 1 mark for explanation of how recursion is used to achieve a total of 161

Question			Expected response	Max mark	Additional guidance
3.	(b)		1. if queue is full then 2. display queue full message 3. else 4. add towelID to queue 5. set rearPointer to rearPointer +1 6. end if	**3**	1 mark for checking for queue overflow 1 mark for adding data 1 mark for increasing pointer
	(c)		The null pointer on the last node of the linked list is replaced with address of the new node, 147. A new node is created to store the data (H128, Hand, 4, 99, 200) which has the pointer value NULL.	**2**	1 mark for each point Note: use of a diagram illustrating updating of pointers and use of additional node is acceptable alternative to complete description
	(d)		Company staff would use the software and provide feedback to the developers on any issues that arise.	**1**	1 mark for use by end users with feedback

Question			Expected response	Max mark	Additional guidance
4.	(a)			**3**	1 mark for (at least) 2 correct actors 1 mark for 6 use cases with interactions 1 mark for appropriate use made of inheritance, extends or includes (between actors or use cases) **OR** 1 mark for appropriate extra actor
	(b)	(i)	`RECORD seatInfo {STRING name, STRING contactNumber, REAL cost, BOOLEAN sold}`	**1**	1 mark for record structure
		(ii)	`DECLARE seatingPlan AS ARRAY OF ARRAY OF seatInfo INITIALLY [[]] <with 13 rows and 4 columns>`	**1**	1 mark for correct 2D array of `seatInfo` records Note: dimensions [12][3] are also acceptable
	(c)		1. loop for rows 0 TO 12 2. loop for columns 0 TO 3 3. if (column = 2 or column = 3) AND (row = 5 or row = 6) then 4. set seatingPlan[row][column] to seatInfo ("Toilet", "Null", 0.00, TRUE) 5. else 6. set seatingPlan[row][column] to seatInfo ("Empty", "Null", 0.00, FALSE) 7. end if 8. end column loop 9. end row loop	**3**	1 mark for correct loops to process the 2D array 1 mark for correctly initialising seats 1 mark for correctly initialising toilets Note: initialisation should make use of record structure that forms each element of the 2D array

Question			Expected response	Max mark	Additional guidance

4.	(d)		SET freeSeatFound TO FALSE
			SET column TO 0
			SET row TO 0
			SET ticketPrice TO 0.00
			
			// Check if seat available
			REPEAT UNTIL freeSeatFound = TRUE
			OR (column = 4 AND row = 13)
			  IF seatingPlan[row]
			  [column].sold = FALSE THEN
			  SET freeSeatFound TO TRUE
			  SET rowFound TO row
			  SET columnFound TO column
			  END IF
			
			  IF column < 3 THEN
			  SET column TO column + 1
			  ELSE
			  SET column TO 0
			  SET row TO row + 1
			  END IF
			END REPEAT
			
			// Offer seat to customer
			IF freeSeatFound = TRUE THEN
			
			  IF rowFound >=0 AND rowFound
			  <=4 THEN
			  SET ticketPrice TO 22.50
			
			  ELSE IF rowFound >=11 AND
			  rowFound <=12 THEN
			  SET ticketPrice TO 75.00
			
			  ELSE
			  SET ticketPrice TO 45.00
			  END IF
			
			  SEND "The ticket price is " &
			  ticketPrice & " proceed with
			  order?" TO DISPLAY
			
			  RECEIVE answer FROM KEYBOARD
			
			  IF answer = <YES> THEN
			  <*store user data and price to record and update sold field of the record*>
			  END IF
			
			ELSE
			  SEND "No seats left" TO
			  DISPLAY
			END IF
				5	1 mark for checking next available seat
					
					1 mark for correct use of 2D array
					
					1 mark calculating correct cost
					
					1 mark for
					• asking user to proceed
					• updating array with correct details
					
					1 mark for
					• correctly displaying not available
					• displaying cost of first available seat
					
					Note: accept assumption stating that prices have already been assigned rather than code/ pseudocode showing the assignment of prices

Question			Expected response	Max mark	Additional guidance
4.	(e)		• One thread checks for availability but before it allocates the seat, another thread does the same – which results in a double-booking • Even if only 1 seat was available, 2 people would believe they have a seat, however the system would have overwritten the final seat allocation • Potential for over-booking the coach	**1**	1 mark for any bullet
	(f)		Adaptive maintenance. Booking management software will need to be adapted to be compatible with platform/device	**1**	1 mark for adaptive with description

[END OF MARKING INSTRUCTIONS]