National
Qualifications
RESOURCE

X816/77/11 Computing Science

Marking Instructions

Please note that these marking instructions have not been standardised based on candidate responses. You may therefore need to agree within your centre how to consistently mark an item if a candidate response is not covered by the marking instructions.

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) Always use positive marking. This means candidates accumulate marks for the demonstration of relevant skills, knowledge and understanding; marks are not deducted.

(c) If a candidate response is not 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) Award marks regardless of spelling, as long as the meaning is unambiguous. This applies to all responses, including code. Award marks as per the detailed marking instructions, regardless of syntax errors, if the intention of the coding is clear.

(e) For questions where candidates are asked to design or write code, a sample response is shown in the detailed marking instructions. This will not be the only valid response. You must use the detailed marking instructions and additional guidance to ensure that you consider alternative approaches and nuances of different programming languages. If in doubt you should refer to your Team Leader.

(f) A correct response can be negated if the candidate includes an extra, incorrect response. For example, if the candidate is asked for two answers for two marks and the candidate gives three, one of which is incorrect, they are awarded one mark.

(g) If a candidate scores through a response and makes a further attempt, you should only mark the further attempt. If no further attempt is made and the original is legible, you should mark the original response.

(h) Where an incorrect response is carried forward and used correctly in a following part of the question, you should give credit for subsequent responses that are correct with regard to the original error. Candidates should not be penalised more than once for the same error.

(i) Only award marks for a valid response to the question asked. Where candidates are asked to:

- **Identify**, **name**, **give** or **state,** they need only name or present in brief form.
- **describe**, they must provide a statement or structure of characteristics and/or features. This will be more than an outline or a list. It may refer to, for example, a concept, process, experiment, situation, or facts, in the context of and appropriate to the question. Candidates must make the same number of factual/appropriate points as there are marks available in the question.
- **explain**, they must relate cause and/or effect and/or make relationships between things clear, in the context of the question or a specific area within the question.
- **write code,** they must write recognisable code, not prose nor a diagram.
- **design,** they must use a design technique appropriate to the problem. Award marks as per the detailed marking instructions, regardless of errors in the exemplification of the technique, if the intention of the design is clear.

(j) In the detailed marking instructions, if a word is underlined then it is essential; if a word is bracketed() then it is not essential. Words separated by / are alternatives

Marking instructions for each question

Section 1

<table>
<tr><th colspan="3">Question</th><th>Expected response</th><th>Max mark</th><th>Additional guidance</th></tr>
<tr><td>1.</td><td></td><td></td><td><table><tr><td>Month</td><td>Hours of Sunshine</td></tr><tr><td>5</td><td>232.8</td></tr><tr><td>7</td><td>218.8</td></tr></table></td><td>2</td><td>1 mark for correct headings

1 mark for correct rows selected</td></tr>
<tr><td>2.</td><td></td><td></td><td>ADD SONG adds a new node at the end of the linked list. The next pointer of the last item in the playlist, the node at memory location 367, will be changed from NULL to point at the new node. The next pointer of the new node will be set to NULL and its previous pointer will be set to 367.</td><td>3</td><td>1 mark for description of ADD SONG option that makes appropriate reference to:
• correct updating of the next pointer of the node at memory location 367
• correct use of NULL as the next pointer of the new node
• correct use of memory location 367 as the previous pointer of the new node</td></tr>
<tr><td>3.</td><td></td><td></td><td>See below.

<pre>CREATE TABLE Order(
   orderID int AUTO-INCREMENT PRIMARY KEY,
   orderDate date NOT NULL,
   customerID varchar(5) NOT NULL,
   FOREIGN KEY (customerID) REFERENCES Customer(customerID)
);</pre></td><td>3</td><td>1 mark for CREATE TABLE with 3 fields and correct data types

1 mark for primary key correct

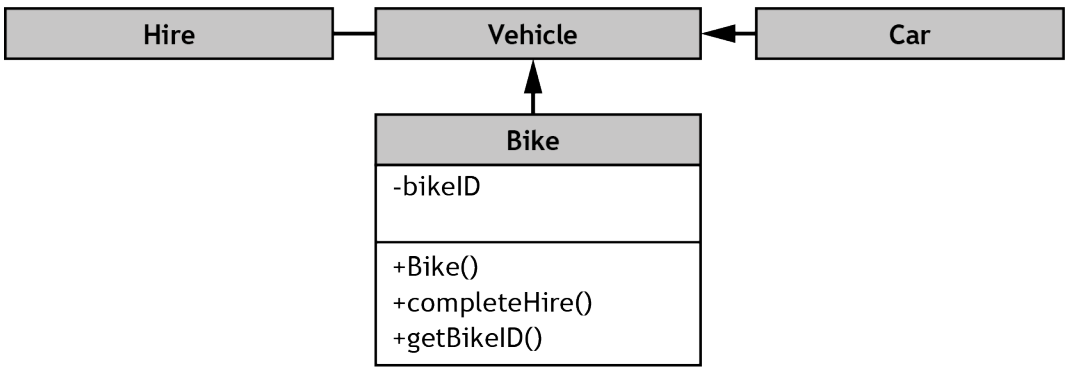
1 mark for foreign key correct</td></tr>
<tr><td>4.</td><td>(a)</td><td></td><td>When the webpage is viewed on a display/device/browser window narrower than 460 pixels.</td><td>1</td><td></td></tr>
<tr><td></td><td>(b)</td><td></td><td>So that the hyperlinks can be displayed vertically rather than horizontally.</td><td>1</td><td></td></tr>
<tr><td></td><td>(c)</td><td></td><td>An additional rule must be added to the media query to tell the browser to hide the main heading

<pre>#mainHeading{display:none}</pre></td><td>1</td><td>Award 1 mark for explanation of need for an additional rule (CSS rule is not required)

Award 1 mark for correct CSS without the explanation.</td></tr>
</table>

Question			Expected response	Max mark	Additional guidance
5.	(a)		IF runners[i].finishTime > runners[i+1].finishTime THEN	**2**	1 mark for comparing i with i+1 1 mark for correct use of data structure Accept alternative comparison: IF runners[i+1] < runners[i] THEN
	(b)		`n` is used to control the size of the inner loop which depends on the number of items that are still to be processed `n` is decreased after a value has been placed in the correct position at the end of the array	**2**	1 mark each for explanation that refers to • control of inner loop which is used to process unsorted items • counting down from sorted end of the array

Section 2

Question			Expected response	Max mark	Additional guidance
6.	(a)		registration = "ABC123" ratePerMile = 7.25 rental empty array of Hire colour = "red" numberHires = 0	**1**	1 mark for all 5 correct Accept rental array is empty
	(b)		Encapsulation means that a property cannot be edited directly. A method must exist in order to update the registration property.	**2**	1 mark for use of encapsulation or use of a private property 1 mark for need for method
	(c)	(i)	Methods from the superclass can be used which avoids the need to code them from scratch	**1**	1 mark for reuse of methods already defined in the superclass
		(ii)	See below.  	**2**	1 mark for `Bike` class with correct properties and methods 1 mark for inheritance indicated correctly]
		(iii)	The `Bike` class inherits the `completeHire()` from the `Vehicle` class. Since the calculation for a bike hire is different from the calculation for a `Car` hire, the inherited calculation must be overridden. Polymorphism is used to replace the `completeHire()` inherited from the super class so that it behaves different when invoked by object belonging to the `Bike` subclass.	**2**	1 mark for `inheritance` of `completeHire()` method 1 mark for need to override inherited method using polymorphism

Question			Expected response	Max mark	Additional guidance

6.	(d)		See below.
			
			`DECLARE maxPosition INITIALLY 0`
			
			`FOR loop FROM 0 TO 124 DO`
			`  IF bikeArray[loop].getNumberHires() >`
			`  bikeArray[maxPosition].getNumberHires() THEN`
			`  SET maxPosition TO loop`
			`  END IF`
			`END FOR`
			`SEND bikeArray[maxPosition].getRegistration() TO DISPLAY`
				3	1 mark for correct use of `bikeArray`
					
					1 mark for use of `getNumberHires()` method
					
					1 mark for use of `getRegistration()` method
					
					Alternative solutions possible

7.	(a)	(i)	The 'registered driver' actor inherits all the characteristics of the 'driver' actor.	**1**	1 mark for inheritance relationship
		(ii)	Extends indicates a conditional use case. In this situation, the 'enter details' use case may lead to the 'invalid details' use case – but it may not.	**1**	1 mark for description that refers to the conditional nature of the relationship
	(b)	(i)	`CarPark` and `Driver` are strong entities `ParkingSpace` and `ParkingRecord` are weak entities	**2**	1 mark for both strong entities correct 1 mark for both weak entities correct
		(ii)	Each parked car in the `ParkingRecord` entity must have a corresponding parking space in the `ParkingSpace` entity so the `ParkingRecord` participation is mandatory. A `ParkingSpace` may not be used so the `ParkingSpace` participation is optional.	**2**	1 mark for mandatory participation of `ParkingRecord` entity 1 mark for optional participation of `ParkingSpace` entity
	(c)		This would produce a list of parking spaces available in car parks 1, 2 and 3.	**1**	

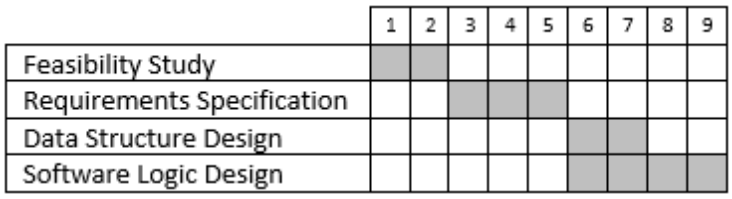
Question			Expected response	Max mark	Additional guidance
7.	(d)	(i)	`HAVING COUNT(*) >= 10`	**1**	
		(ii)	`AND carParkName IN ('Rose Street', 'Bank Street')`	**1**	
	(e)	(i)	A: NOT carParkName = “Ness Walk” B: sum(cost) >\ C: sum(cost) x 2	**3**	1 mark for each correct
	(e)	(ii)	See below.	4	1 mark for initialisation of low, high and midpoint 1 mark for suitable conditional loop (accept upper<low) 1 mark for correct updating of low and high 1 mark for correct use of data structure given in question

```
1.  set low = 0
2.  set high = 266
3.  set found = false
4.  enter seachSpaceID
5.  start conditional loop
6.      set mid = (low + high)/2
7.      if spaceList[mid].spaceID < searchSpaceID then
8.          set low = mid + 1
9.      end if
10.     if spaceList[mid].spaceID > searchSpaceID then
11.         set high = mid - 1
12.     end if
13.     if spaceList[mid].spaceID = searchSpaceID then
14.         set found = true
15.     end if
16. loop until (found = true) or (low > upper)
17. if found = false then
18.         display ‘not found’ message
19. else
20.         display spaceList[mid].income
21. end if
```

Question			Expected response	Max mark	Additional guidance
8.	(a)	(i)	`<form method="post" action="requestConfirm.php" >`	**2**	1 mark for correct action 1 mark for correct method
		(ii)	`Date` `<input type = "text" value = "yyyy-mm-dd">`	**1**	1 mark for `input` element with correct use of the `value` attribute
	(b)	(i)	Session variables allow values stored in variables to persist across several pages of the website. This means that the values can be used on multiple pages of the site. Session variables have been used on this page to store details of the car (make and model), customer (name and email).	**2**	1 mark for use made of session variables 1 mark for details of values being stored in session variables

		(ii)	`//assign connection details`
			`$servername = "serv1a";`
			`$username = "root";`
			`$password = "AQS20";`
			`$database="TestDrive";`
			
			`// create connection`
			`$conn = mysqli_connect($servername, $username, $password,$database);`
				2	1 mark for assignment of connection details to server-side variables
					
					1 mark for assignment of `$conn` that makes use of `mysqli_connect` function with 4 parameters

	(c)	(i)	`INSERT INTO ClientTD48 (customerName, startDate, email, requestStatus) VALUES ('$customerName','$startDate','$email',"pending");`	**2**	1 mark for `INSERT` query with correct list of fields Note: `id` field should not be listed since it is an auto increment field 1 mark for list of values to match field sequence
	(c)	(ii)	`$rowcount = mysqli_num_rows($result);`	**2**	1 mark for correct use of `$rowcount` 1 mark for correct use of `$result` as function argument

Question			Expected response	Max mark	Additional guidance
8.	(c)	(iii)	1. set pending = 0 2. start conditional loop while there are still query results to process 3. if pending < 5 then 4. update requestStatus field of current record to 'successful' 5. else 6. update requestStatus field of current record to 'unsuccessful' 7. end if 8. add 1 to pending 9. end conditional loop	**3**	1 mark for ensuring that only 5 records are successful 1 mark for updating successful records 1 mark for updating unsuccessful records Alternative solutions possible
	(d)		This could be achieved using the `session_destroy()` statement.	**1**	
9.	(a)	(i)	 1 mark for Feasibility Study, Requirements Specification and Data Structure Design happening sequentially. overlapping design stages and overlapping implementation and integrative testing. 1 mark for overlapping Data Structure Design and Software Logic Design.	**2**	
		(ii)	If the cost of creating the software upgrade is prohibitively high, it will not be feasible to continue with the development work.	**1**	

	(b)	(i)	`SET lifts TO [ [ ] ] * 8`
			`FOR count FROM 0 TO 7 DO`
			`SET lifts[ count ] TO [""] * 4`
			`END FOR`
			
			OR
			
			`DECLARE lifts AS ARRAY OF ARRAY OF CHARACTER`
			`INITIALLY < 4 x 8 array, all set to blank >`
				2	1 mark for dimensions
					
					1 mark for data type
					
					Dimensions of 2D array must be indicated.
					
					Data type must be indicated explicitly as CHARACTER or implied using "" for null string.

		(ii)	`SET lifts [2] [0] TO "A"`
			`SET lifts [7] [1] TO "S"`
			`SET lifts [2] [2] TO "D"`
			`SET lifts [5] [3] TO "D"`
				1	1 mark for correct assignment of all values
					
					Accept answers where the bottom row is numbered from 0

Question			Expected response	Max mark	Additional guidance
9.	(b)	(iii)	See below.	**4**	

```
FOR row FROM 0 TO 7 DO
  FOR column FROM 0 TO 3 DO
    IF lifts [row] [column] <> "" THEN
        IF lifts [row] [column] = "S" THEN
           SET distance[column] TO floor - row
        ELSE IF direction = "A" AND row > floor THEN
          SET distance[column] TO floor - row
        ELSE IF direction = "D" AND row < floor THEN
          SET distance[column] TO floor - row
        ELSE
          SET distance[column] TO 99
        END IF
    END IF
  END FOR
END FOR
```

1 mark for nested loop to process 2-D array

1 mark for finding element with lift in each column

1 mark for deciding if lift will not be called (assign 99 to distance)

1 mark for calculating and storing difference if lift can be called

		(iv)	See below.	**2**	

```
FOR EACH value FROM distance DO
   IF value < 0 THEN
       SET value TO value * -1
   END IF
END FOR EACH
SET min TO 0
FOR i FROM 1 TO 3 DO
   IF distance[i] < min THEN
       SET min TO i
   END IF
END FOR
SEND "Lift " & i + 1 TO DISPLAY
```

Note: a pre-defined function may be used to find magnitude and ignore sign of each value. Or negative values can be made positive inside finding minimum algorithm.

1 mark for dealing with negative values

1 mark for finding correct lift

	(c)		Final testing could take place in an empty building overnight.	**1**	1 mark for referring to use in a real-life context, but without people in the lifts.

[END OF MARKING INSTRUCTIONS]