H

National
Qualifications
2022

2022 Computing Science

Higher

Finalised Marking Instructions

© Scottish Qualifications Authority 2022

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

The information in this document may be reproduced in support of SQA qualifications only on a non-commercial basis. If it is reproduced, SQA must be clearly acknowledged as the source. If it is to be reproduced for any other purpose, written permission must be obtained from permissions@sqa.org.uk.

SQA

General marking principles for 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.

(e) Award marks as per the detailed marking instructions, regardless of minor syntax errors, if the intention of the coding is clear.

(f) 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.

(g) If a candidate scores through their entire response to a question 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 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 - Software design and development, and Computer systems

Question			Expected response	Max mark	Additional guidance
1.			• -2^{31} OR -2 147 483 648 • $2^{31} - 1$ OR 2 147 483 647	**2**	1 mark for each bullet
2.			• sign bit: 1 • 111 0000 1111 0101 • exponent: 0000 1011	**3**	1 mark for each bullet
3.			**Vector** • individual objects/attributes can be edited/moved independently **OR** • objects can be layered **Bitmap** • edit individual pixels	**2**	1 mark from vector 1 mark from bitmap Maximum 2 marks Reference to moving shapes/ rectangles is in the stem. Must reference objects, attributes or layers.
4.			• larger cache stores more instructions/data • more cache hits/fewer cache misses • reduces fetches from main memory	**2**	1 mark for each bullet Maximum 2 marks
5.	(a)		A= [5,7,0,0,0,0] B= [5,7,7,0,0,0] C= [5,7,7,7,0,0]	**3**	1 mark for each line
	(b)		sequence[n]=sequence[n-1] (1 mark) +sequence[n-2] (1 mark)	**2**	

Question			Expected response	Max mark	Additional guidance
6.	(a)		Design does not account for the same score(shortest time).	**1**	
	(b)	(i)	• record declaration • 6 appropriate fields within a record	**2**	1 mark for each bullet RECORD player IS {STRING uniqueID, INTEGER score, REAL minutes, INTEGER speed, INTEGER strength, INTEGER agility} Do not award second mark if an incorrect data type has been applied to a field.
		(ii)	• array of (10 000) records • using data structure from (i)	**2**	1 mark for each bullet DECLARE allPlayers AS array of player INITIALLY [NULL] *10 000
	(c)		• loop for length of array/10 000 iterations • IF current minutes >500 • increase current speed by 3 • correct use of array name and field names	**4**	1 mark for each bullet For 'current' must reference array position. Award mark for 'current' if array is referenced without record structure.

```
FOR index FROM 0 TO 9 999 DO
    IF allPlayers[index].minutes>500 THEN
        allPlayers[index].speed= allPlayers[index].speed+3
    END IF
END FOR
```

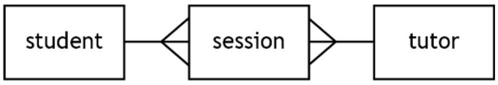
	(d)		• modules can be reused. • each part of the solution can be tested independently. • errors are easier to locate. • different programmers can code different modules (independently). • variables only in memory during execution of subroutine. • limits accessibility of local variables.	**2**	1 mark for each bullet Maximum 2 marks
	(e)		• ensures that all (functional) requirements of the program are tested.	**1**	

Question			Expected response	Max mark	Additional guidance
7.	(a)		• shots (per hole) >= 1 **OR** shots (per game)>=9 • exactly 9 scores for each player • number of holes-in-one >=0 **OR** <=9 • number of holes won is >= 0 **OR** <=9	**2**	1 mark for each bullet Maximum 2 marks
	(b)		In for both: playerAName playerBName playerAscores() playerBscores()	**2**	Any additional inputs or outputs for either step then do not award the mark for that step. Indication of IN must be shown.
	(c)		• initialise/create count for A and B • IF statement for A win • IF statement for B win • increment count for A and B • output for A/B win • condition for draw	**6**	1 mark for each bullet Do not double penalise IF statement. Answer must make two separate comparisons (IFs) to account for possible drawn holes.
			Design: winsA = 0 winsB = 0 loop for 9 holes if playerAscores()<playerBscores() then winsA = winsA + 1 else if playerBscores()<playerAscores() then winsB = winsB + 1 end if end loop if winsA>winsB then display playerAname has won most holes else if winsB>winA then display playerBname has won most holes else if winsA=winsB then display game is drawn end if		
	(d)	(i)	Description of additional parameter/ mismatch in the number of (actual vs formal) parameters.	**1**	
		(ii)	Call to function with correct parameter.	**1**	holesInOne(playerAScores)
	(e)		Declaration inside subroutine **(1 mark)** then only accessible within that subroutine (therefore local) **(1 mark)**. **OR** Declaration outside subroutines **(1 mark)** then accessible throughout entire program (therefore global) **(1 mark)**.	**2**	1 mark for description of scope relating to both subroutine and entire program.

Question			Expected response	Max mark	Additional guidance
8.	(a)		• generate hint • store letters guessed • check entered letter is correct • identify/allocate letter to correct box • count number of guesses • decide if whole word is correct • check entered letter against stored letters	**2**	1 mark for each bullet Maximum 2 marks Do not award input/output marks.
	(b)		• Open & close file • Read file into data structure • Generate random number • Assign word at position random number to chosenWord variable	**4**	1 mark for each bullet
	(c)	(i)	Assignment of first character to `hintOne`.	**1**	hintOne= left(chosenWord, 1) hintOne= mid(chosenWord, 1, 1) hintOne= chosenWord[0] hintOne= chosenWord[0:1] hintOne= chosenWord[:1] put char 1 of chosenWord into hintOne
		(ii)	• use of length function to generate random number for length of chosenWord • assign letter at position identified to hintTwo	**2**	1 mark for each bullet randomPos = random(1, len(chosenWord)) hintTwo = mid(chosenWord, randomPos,1) hintTwo=chosenWord[random.randInt(0,len(chosenWord)-1)]
	(d)		• conditional loop with terminating condition and input within conditional loop • loop to traverse the array • decide if letter is in usedLetters array • display error message if letter in usedLetters • valid letter is added to array • number of letters incremented	**6**	1 mark for each bullet Allow loop for length of array of usedLetters/26 iterations or for numLetters. Only award last two marks if placed in section of code when letter is not found.

```
DECLARE found INITIALLY false
REPEAT
   RECEIVE letter FROM KEYBOARD
      FOR index FROM 0 TO usedLetters-1 DO
         IF usedLetters[index]=letter THEN
            SET found TO true
            SEND ‘Letter already used. Please re-enter’
         END IF
      END FOR
UNTIL found=false
SET usedLetters[numLetters] TO letter
SET numLetters TO numLetters +1
```

Section 2 – Database design and development

Question			Expected response	Max mark	Additional guidance
9.	(a)		• (Create a query to) search for the contact details for a particular tutor or student • (Create a query to) count the number of tutor sessions on a particular date • (Create a query using SUM to) calculate the total tutor fees that each staff member has been paid • (Create a query using MIN) to calculate the tutor who charges the lowest hourly rate • Create/update/delete records to store details of student, staff and tutor sessions	**1**	1 mark for any one bullet
	(b)			**2**	1 mark for each correct relationship
10.			• MIN & MAX functions are wrong way round OR ALIAS assigned to wrong function • no wildcard operator both sides of 1 • no GROUP BY squad	**3**	Accept wildcard (* or %) eg WHERE squad LIKE "%1%" Wildcard operator on the right can be for a single character.

<table>
<tr><th colspan="3">Question</th><th>Expected response</th><th>Max mark</th><th>Additional guidance</th></tr>
<tr><td><b>11.</b></td><td>(a)</td><td></td><td>• Referral, Customer
• specialist Like “*Eye Clinic*”
• (AND) referralDate Like "*/04/2022"

<table><tr><td>Tables(s)</td><td>Referral, Customer</td></tr><tr><td>Search criteria</td><td>specialist Like "%Eye Clinic%"
AND referralDate Like "%/04/2022"

OR

specialist Like "*Eye Clinic*"
AND referralDate Like "*/04/2022"</td></tr></table></td><td><b>3</b></td><td>1 mark for each bullet

In design accept other expressions for like eg =

Access allows:

referralDate Like "??/04/2022"

SQL allows:
referralDate Like "_ _/04/2022"

Ignore any content in Grouping.</td></tr>
<tr><td></td><td>(b)</td><td>(i)</td><td>• aggregate function AVG on loyaltyPoints
• alias and table</td><td><b>2</b></td><td>1 mark for each bullet

<code>SELECT avg(loyaltyPoints) as ‘Average Points’</code>

<code>FROM Customer</code></td></tr>
<tr><td></td><td></td><td>(ii)</td><td>• FROM Customer, Optician, avgPointsQuery

• equi-join: WHERE Customer.opticianID = Optician.optiicanID

• loyaltyPoints > ‘Average Points’

• loyaltyPoints descending</td><td><b>4</b></td><td>1 mark for each bullet

<code>SELECT forename, surname, loyaltyPoints, opticianName</code>

<code>FROM Customer, Optician, avgPointsQuery</code>

<code>WHERE Customer.opticianID = Optician.optiicanID AND loyaltyPoints > ‘Average Points’</code>

<code>ORDER BY loyaltyPoints DESC</code></td></tr>
</table>

<table>
<tr><th colspan="3">Question</th><th>Expected response</th><th>Max mark</th><th>Additional guidance</th></tr>
<tr><td><b>12.</b></td><td>(a)</td><td></td><td>• saleID is able to uniquely identify the record
• The primary keys from the other two tables would not uniquely identify a car for sale.
• There would be no customer number or salesPersonRef until a sale had been made and the values in a compound key cannot be left empty.</td><td><b>1</b></td><td>1 mark for any one bullet</td></tr>
<tr><td></td><td>(b)</td><td></td><td><table><tr><td>Field(s) and calculation(s)</td><td>year,
min(askingPrice) AS ‘Cheapest Price’</td></tr><tr><td>Search criteria</td><td>year=2017 OR
year=2018
AND Sold = ‘No’</td></tr><tr><td>Grouping</td><td>year</td></tr></table></td><td><b>3</b></td><td>1 mark for each row

Allow use of minimum in design of query.

Correct syntax is not required in design.</td></tr>
<tr><td></td><td>(c)</td><td></td><td>• fields and alias
• calculation (askingPrice - soldPrice)
• tables and equi-join
• sold = ‘Yes’;</td><td><b>4</b></td><td>1 mark for each bullet

<code>SELECT carReg, salesPersonName, askingPrice, soldPrice, (askingPrice-soldPrice) AS [Price Difference]</code>

<code>FROM CarSale, SalesPerson</code>

<code>WHERE CarSale.salesPersonRef = SalesPerson.salesPersonRef and sold = ‘Yes’;</code>

<code>Sold=’Yes’ could be established several other ways e.g. soldPrice is not null</code></td></tr>
<tr><td></td><td>(d)</td><td></td><td>• UPDATE CarSale and WHERE mileage <=10000
• SET askingPrice = askingPrice * 1.10</td><td><b>2</b></td><td>1 mark for each bullet

Award 0 marks for first bullet if there are any additional SQL clauses.

<code>UPDATE CarSale</code>
<code>SET askingPrice = askingPrice * 1.10</code>
<code>WHERE mileage <=10000;</code>

<code>Or</code>
<code>SET askingPrice=askingPrice +askingPrice/10</code></td></tr>
</table>

Section 3 – Web design and development

Question			Expected response	Max mark	Additional guidance
13.			A = nav B = section	**2**	For B accept article/div
14.			• Fields - title,description, type, start date, end date • radio buttons/option controls for Job Type • submit button • required for all fields except end date	**4**	1 mark for each bullet Validation can include * or labels.
15.	(a)		• To get to the descendant pages the user has to navigate through the previous instructor pages. • No FAQ page	**2**	1 mark for each bullet
	(b)		• form input(s) have no labels • No obvious navigation structure/ navbar/orphan page	**2**	1 mark for each bullet

	(c)		• h1,h2,h3 grouping selector
			• h2 with 10px padding rule
				2	1 mark for each bullet
					
					h1,**h2,h3** {font-family: Verdana; text-align: center;}
					h2 {padding: 10;}
					or h2{padding:10 10 10 10;}
					10px optional in all cases.

	(d)		• the first unordered list will be styled yellow OR Beginners/Refresher (li elements) are yellow. • the li elements that are descendants of ul and ol will be blue OR details of the courses ie cost, min lessons, lesson length will be blue.	**2**	1 mark for each bullet
	(e)		• test that the website functions on different browsers • test that the website functions on different devices	**2**	1 mark for each bullet

Question			Expected response	Max mark	Additional guidance

16.	(a)		• margin-top: 30px
			• padding-top: 20px
				2	1 mark for each bullet
					`margin{30 0 0 0;}`
					`padding{20 0 0 0;}`

	(b)		• a:hover selector • background-color is black, color is white	**2**	1 mark for each bullet `nav ul li` **`a:hover`** `{background-color:black; color: white;}`
	(c)		An element with an inline property only takes up the width that is needed.	**1**	
	(d)	(i)	• onclick event • correct displayHelp function call	**2**	1 mark for each bullet `<img src="question.jpg"` **`onclick="displayHelp()`**`">` Parameter 'this' may be included.
		(ii)	Incorrect ID 'display' is used as a parameter in the function. **OR** Correct ID 'help' should be used as a parameter in the function.	**1**	
	(e)		Add **required** to electricity input.	**1**	`<input type="number" name="electricity"` **`required`**`>`

[END OF MARKING INSTRUCTIONS]