National
Qualifications
2026

2026 Computing Science

Higher

Question Paper Finalised Marking Instructions

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

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

Qualifications Scotland
Teisteanasan Alba

© Qualifications Scotland 2026

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. 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) If a candidate puts a score 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.

(g) 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.

(h) 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.

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

Marking instructions for each question

SECTION 1- SOFTWARE DESIGN AND DEVELOPMENT, AND COMPUTER SYSTEMS

<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>(a)</td><td></td><td>-86</td><td>1</td><td></td></tr>
<tr><td></td><td>(b)</td><td></td><td>10</td><td>1</td><td></td></tr>
<tr><td>2.</td><td>(a)</td><td></td><td>Sign: 0
Mantissa: 111 0000 0000 0000
Exponent: 0000 0001</td><td>3</td><td></td></tr>
<tr><td></td><td>(b)</td><td></td><td>increase number of bits assigned to mantissa and decrease number of bits assigned to exponent</td><td>1</td><td></td></tr>
<tr><td>3.</td><td></td><td></td><td><table><tr><th>Formal</th><th>Actual</th></tr><tr><td>price</td><td>originalPrice</td></tr><tr><td>percentage</td><td>discountRate</td></tr></table></td><td>2</td><td>Award 2 marks for correct pair.

Award 1 mark for one correctly identified formal/actual.</td></tr>
<tr><td>4.</td><td></td><td></td><td>• fewer cache hits
• fewer frequently used instructions stored
• more fetch executes required (to slower main memory)</td><td>2</td><td>Award 1 mark for each bullet. Maximum 2 marks.

Accept inverse.</td></tr>
<tr><td>5.</td><td></td><td></td><td>• loop for each character
• identifying position of @ character
• set domainName to all characters after the "@" sign</td><td>3</td><td>Accept use of predefined functions for bullets 1 and 2.</td></tr>
</table>

Question			Expected response	Max mark	Additional guidance
6.	(a)		• pressure must be a real number or integer • required pressure input(s) must be > 0 • pressure input cannot exceed the maximum (100) • maximum pressure > 0	**1**	Award **1 mark** for any one bullet.
	(b)		• (domain) name of the site/IP address • name of organisation issued to • the public key • date on which the certificate is issued • the name of the issuing Certificate Authority • the digital signature (of the issuing Certificate Authority) • Serial Number	**1**	Award **1 mark** for any one bullet.
	(c)		**Digital certificate:** • proves identity of the sender/website • allows encryption/security **Digital signature:** • proves message has not been altered/integrity of the message	**2**	
	(d)		**Traffic is kept moving by:** • using a variety of information from cameras and sensors, along with control of traffic signals • navigation systems rerouting cars away from busy traffic/towards more eco friendly routes	**1**	Award **1 mark** for any one bullet.

<table>
<tr><th colspan="3">Question</th><th>Expected response</th><th>Max mark</th><th>Additional guidance</th></tr>
<tr><td>7.</td><td>(a)</td><td></td><td>

- Line 6 `totals` = [0,0,0]
- both updates of `currentDrink`
- Line 8 `currentDrinkTotal` = 4
- Line 9 `totals` = [0,4,0]

<table>
<tr><th>Line Number</th><th>index</th><th>totals</th><th>currentDrink</th><th>currentDrinkTotal</th></tr>
<tr><td>5</td><td>0</td><td></td><td></td><td></td></tr>
<tr><td>6</td><td></td><td>[0,0,0]</td><td></td><td></td></tr>
<tr><td>7</td><td></td><td></td><td>Tea</td><td></td></tr>
<tr><td>8</td><td></td><td></td><td></td><td>5</td></tr>
<tr><td>9</td><td></td><td>[5,0,0]</td><td></td><td></td></tr>
<tr><td>5</td><td>1</td><td></td><td></td><td></td></tr>
<tr><td>6</td><td></td><td>[0,0,0]</td><td></td><td></td></tr>
<tr><td>7</td><td></td><td></td><td>Coffee</td><td></td></tr>
<tr><td>8</td><td></td><td></td><td></td><td>4</td></tr>
<tr><td>9</td><td></td><td>[0,4,0]</td><td></td><td></td></tr>
</table>

</td><td>4</td><td></td></tr>
<tr><td></td><td>(b)</td><td></td><td>

- Line 6
- move the declaration outside of the loop/swap lines 5 and 6

</td><td>2</td><td></td></tr>
<tr><td></td><td>(c)</td><td></td><td>

- stops execution when a variable changes/meets condition
- allowing variable values to be compared to expected values/allows variable to be inspected

</td><td>2</td><td></td></tr>
</table>

Question			Expected response	Max mark	Additional guidance
8.	(a)		• use of mod function/operator • arguments (number, total) in order and assignment to result	**2**	Example answer: `SET result TO number MOD total`
	(b)		• use of function name `checkHarshad` with assignment to `finalResult` • parameter `checkNo` passed into function	**2**	`finalResult = checkHarshad(checkNo)`
	(c)	(i)	It is declared/used/accessible inside the function/lines 1 – 12.	**1**	
		(ii)	The variables are only in memory when the function is being executed.	**1**	
9.	(a)	(i)	3	**1**	
		(ii)	• only stores/displays the position of the last (thriller) • does not display multiple positions	**1**	Position is set to 0 (and will be returned as 0 if none found).
		(iii)	• move line 8 to after line 5/display within if statement • create an array/concatenate a string to store each position value • remove line 8 and change line 5 to display `Index`	**1**	Award **1 mark** for any one bullet.
	(b)		• initialise and update number of movies • loop for entire array • if current genre = "Drama" and current type = "TV Show" • open/create file • write the number of movies to file	**5**	
	(c)		Can represent a larger amount of characters/languages.	**1**	
	(d)		Can build profile of sites that are visited/target advertisements.	**1**	

Question			Expected response	Max mark	Additional guidance

10.	(a)	(i)	• record declaration
			• 4 appropriate fields within a record
			
			Example answer:
			`RECORD book IS {STRING title,`
			`STRING author, INTEGER year, REAL buyingPrice }`
				2	If data types are indicated they must be appropriate.

		(ii)	• array of 2000 elements
			• using data structure from (i)
			
			Example answer:
			`DECLARE books AS ARRAY OF book INITIALLY []*2000`
				2	Allow indexing from 1 to 2000.
					
					Award **1 mark** for
					`books = [Book()]*2000`
					`books = [Book]*2000`

	(b)		Step1 OUT: • books() Step 3 IN: • books(), searchTitle, searchAuthor Step 4 IN: • foundPosition, books()	**3**	Array passed out in step 1 does not need to match part(a) but must avoid confusion with specific fields. For each step the mark is awarded for the correct parameter(s) with no additional parameters. If candidate passes out parallel arrays/fields: • do not award step 1 • step 3 IN: • year()/array().year, title()/array().title, author()/array().author • step 4 IN: year()/array().year

<table>
<tr><th colspan="3">Question</th><th>Expected response</th><th>Max mark</th><th>Additional guidance</th></tr>
<tr><td>10.</td><td>(c)</td><td></td><td>

- suitable initialisation and update of foundPosition
- suitable initialisation and update firstYear
- if condition = searchTitle and condition = searchAuthor
- if condition < firstYear with loop to iterate through all elements of array
- array name matching to (a)
- use of field name(s) in IF condition (title, author, year)

Example answer:

```
SET foundPosition TO -1
SET firstYear TO <any suitable large value>

FOR x FROM 0 to length(books)-1 DO
  IF books[x].title = searchTitle AND books[x].author =
      searchAuthor AND books[x].year < firstYear THEN
      SET foundPosition TO counter
      SET firstYear TO books[x].year
    END IF
END FOR
```

</td><td>6</td><td>

Award **1 mark** for each bullet.

At bullet 2,3,4 do not penalise if parallel arrays are used.

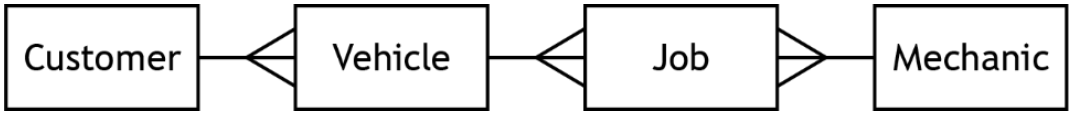
If parallel arrays are used do not award bullets 5 and 6.
Array index must be used correctly.

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

SECTION 2 — DATABASE DESIGN AND DEVELOPMENT

<table>
<tr><th colspan="3">Question</th><th>Expected response</th><th>Max mark</th><th>Additional guidance</th></tr>
<tr><td>11.</td><td></td><td></td><td>

- Customer – Vehicle (1:M)
- Mechanic – Job (1:M)
- Vehicle – Job (1:M)



</td><td>3</td><td></td></tr>
<tr><td>12.</td><td></td><td></td><td>

Season	Shortest Route
Autumn	9
Summer	8
Spring	4

</td><td>2</td><td>Three correct records.
Correct sort order.</td></tr>
<tr><td>13.</td><td></td><td></td><td>

- SUM
- calculation and alias
- use of wildcards in search criteria

Example answer:

```
SELECT SUM(billedCost – expenses) as [Total profit]
FROM Job
WHERE jobDescription LIKE '%paint%'
```

</td><td>3</td><td>Note: Access will use *

Accept `SUM(billedCost)` `SUM(expenses)` for bullet 1/2.</td></tr>
<tr><td>14.</td><td>(a)</td><td></td><td>eventid, archerID</td><td>1</td><td>Both required.</td></tr>
<tr><td></td><td>(b)</td><td></td><td>

- FROM clause includes 'Overall Average'
- points > AveragePoints
- Archer, Event, Result tables with joins

Example answer:

```
SELECT forename, surname, eventName, points
FROM Archer, Event, Result, OverallAverage
WHERE Archer.archerID = Result.archerID
AND Event.eventID = Result.eventID
AND points > AveragePoints;
```

</td><td>3</td><td></td></tr>
<tr><td></td><td>(c)</td><td></td><td>

- COUNT aggregate function
- three tables
- type = "Lawn" and style = "Left-handed"

Field(s) and Calculations	COUNT(*)
Table(s)	Event, Result, Archer
Search Criteria	type = "Lawn" AND style = "Left-handed"
Grouping	
Sort Order	

</td><td>3</td><td></td></tr>
</table>

Question			Expected response	Max mark	Additional guidance
15.	(a)		**Create a query to:** • use MAX/search for the highest cost of a boat trip (and display the name of the customer who booked that trip) • use MAX/search for the highest number of booking (and display the name of the captain) • use COUNT/calculate the number of trips on a specific date	**2**	Award **1 mark** for any one bullet. Maximum **2 marks**.

	(b)		• correct UPDATE syntax with criteria
			• calculation and set new value
				2	Award **0 marks** for first bullet if SQL clauses are in the wrong order.
					
					`UPDATE Booking`
					`SET price = price-15`
					`WHERE partySize < 10`
					`AND timeOut < "12:00"`
					
					Accept `12,#12:00#` or no speech marks/square brackets.

	(c)		• boatName field and alias
			• AVG(passengers) with tables and equijoin
			• GROUP BY boatName/boatID
			
			Example answer:
			`SELECT boatName, AVG(passengers) AS [Average Passengers]`
			`FROM Boat, BoatBooking`
			`WHERE Boat.boatID = BoatBooking.boatID`
			`GROUP BY boatName`
				3	

	(d)		• missing MIN around passengers • missing join between Booking and BoatBooking tables • missing ORDER BY MIN(passengers)/[Least passengers] DESC	**3**	

SECTION 3 — WEB DESIGN AND DEVELOPMENT

Question			Expected response	Max mark	Additional guidance
16.	(a)		• a form to create an account • a form to sign into an account • JavaScript to view the details of a selected player	**2**	
	(b)		• the Manager Page doesn't link to the Staff Page/only accessible through the Player page • the European Page doesn't link to the Tickets Page/should not be accessed through the shop page	**2**	
17.			• name, address, email and mobile number all being required • dropdown menu or radio buttons for favourite type of photography • submit button	**3**	Accept firstname/surname. Accept */required for bullet 1.
18.	(a)	(i)	function displayNew(my_image){ my_image.src =' ../images/newBadge.png'; }	**1**	Mark can be awarded if function correct with or without relative file path.
		(ii)	• calling `displayOld` using onmouseover • calling new function using onmouseout	**2**	`<img src="newBadge.png" onmouseover="displayOld(this)" onmouseout="displayNew (this)">` New function for bullet 2 must match a(i).
	(b)		• `header h1` • `display: inline`	**2**	Accept `float:left` for bullet 2. `header h1{display: inline;}`
	(c)		• `header, footer {margin-top: 10px;}` • `header {height: 200px;} footer {height: 40px;}` • `#stadiumImg, #infoSection{ float: left; width: 590px; height: 600px }`	**3**	
	(d)		• dropdown list allows multiple selection • dropdown menu takes up less space on the page	**2**	Accept inverse.

Question			Expected response	Max mark	Additional guidance

19.	(a)		• `#leftSection {` **`margin-right:10px;`**`}`
			
			• `#rightSection {` **`padding: 20px;`**`}`
				2	Accept `margin-left` added to `#right`
					
					`padding:20 20 20 20;`
					
					`padding-left:20px;`
					`padding-top:20px;`
					`padding-right:20px;`
					`padding-bottom:20px;`

	(b)		• descendant (selector) • used to style the unordered lists within the nav element	**2**	
	(c)		• `select` element copied with `multiple` removed • `size = "3"` added	**2**	`<select name = "volunteer" size = "3">`
	(d)		**Persona:** • (a tester) playing the role of a typical end user **Test case:** • a set of actions executed (by a tester) to test a particular feature/function	**2**	

[END OF MARKING INSTRUCTIONS]