

Cambridge IGCSE™

COMPUTER SCIENCE**0478/22**

Paper 2 Algorithms, Programming and Logic

February/March 2025**MARK SCHEME**Maximum Mark: 75

Published

This mark scheme is published as an aid to teachers and candidates, to indicate the requirements of the examination. It shows the basis on which Examiners were instructed to award marks. It does not indicate the details of the discussions that took place at an Examiners' meeting before marking began, which would have considered the acceptability of alternative answers.

Mark schemes should be read in conjunction with the question paper and the Principal Examiner Report for Teachers.

Cambridge International will not enter into discussions about these mark schemes.

Cambridge International is publishing the mark schemes for the February/March 2025 series for most Cambridge IGCSE, Cambridge International A and AS Level components, and some Cambridge O Level components.

This document consists of **21** printed pages.

PUBLISHED**Generic Marking Principles**

These general marking principles must be applied by all examiners when marking candidate answers. They should be applied alongside the specific content of the mark scheme or generic level descriptions for a question. Each question paper and mark scheme will also comply with these marking principles.

GENERIC MARKING PRINCIPLE 1:

Marks must be awarded in line with:

- the specific content of the mark scheme or the generic level descriptors for the question
- the specific skills defined in the mark scheme or in the generic level descriptors for the question
- the standard of response required by a candidate as exemplified by the standardisation scripts.

GENERIC MARKING PRINCIPLE 2:

Marks awarded are always **whole marks** (not half marks, or other fractions).

GENERIC MARKING PRINCIPLE 3:

Marks must be awarded **positively**:

- marks are awarded for correct/valid answers, as defined in the mark scheme. However, credit is given for valid answers which go beyond the scope of the syllabus and mark scheme, referring to your Team Leader as appropriate
- marks are awarded when candidates clearly demonstrate what they know and can do
- marks are not deducted for errors
- marks are not deducted for omissions
- answers should only be judged on the quality of spelling, punctuation and grammar when these features are specifically assessed by the question as indicated by the mark scheme. The meaning, however, should be unambiguous.

GENERIC MARKING PRINCIPLE 4:

Rules must be applied consistently, e.g. in situations where candidates have not followed instructions or in the application of generic level descriptors.

GENERIC MARKING PRINCIPLE 5:

Marks should be awarded using the full range of marks defined in the mark scheme for the question (however; the use of the full mark range may be limited according to the quality of the candidate responses seen).

GENERIC MARKING PRINCIPLE 6:

Marks awarded are based solely on the requirements as defined in the mark scheme. Marks should not be awarded with grade thresholds or grade descriptors in mind.

Annotations guidance for centres

Examiners use a system of annotations as a shorthand for communicating their marking decisions to one another. Examiners are trained during the standardisation process on how and when to use annotations. The purpose of annotations is to inform the standardisation and monitoring processes and guide the supervising examiners when they are checking the work of examiners within their team. The meaning of annotations and how they are used is specific to each component and is understood by all examiners who mark the component.

We publish annotations in our mark schemes to help centres understand the annotations they may see on copies of scripts. Note that there may not be a direct correlation between the number of annotations on a script and the mark awarded. Similarly, the use of an annotation may not be an indication of the quality of the response.

The annotations listed below were available to examiners marking this component in this series.

Annotations

Annotation	Meaning
	Correct point
	Incorrect point
	Follow through
	Repetition
	Ignore
	Benefit of doubt given
	Content of response too vague
	Not answered question
	Omission
	Section not relevant

Annotation	Meaning
	Section incorrect
Highlighter	Highlights part of the answer or shows structure of complex answers
	Page or response seen by examiner
	AO2 mark
	AO3 mark
	Not enough
	Required item one
	Required item two
	Required item three
	Correct awarding one mark
	Correct awarding two marks
	Correct awarding three marks
	Correct awarding four marks
	Correct awarding five marks
	Correct awarding six marks
	Correct awarding seven marks
	Correct awarding eight marks
	Correct awarding nine marks

Mark scheme abbreviations

/ separates alternative words / phrases within a marking point

// separates alternative answers within a marking point

underline actual word given must be used by candidate (grammatical variants accepted)

max indicates the maximum number of marks that can be awarded

() the word / phrase in brackets is not required, but sets the context

Note: No marks are awarded for using brand names of software packages or hardware.

Question	Answer	Marks
1	C	1

Question	Answer	Marks
2	B	1

Question	Answer	Marks												
3	One mark for each correct line <table><thead><tr><th>Stage</th><th>Description</th></tr></thead><tbody><tr><td>testing</td><td>identifying of the problem and requirements</td></tr><tr><td>analysis</td><td>reviewing the final solution to suggest further developments</td></tr><tr><td>coding</td><td>making sure the program code works as expected</td></tr><tr><td>design</td><td>using structure diagrams, flowcharts and pseudocode to plan the solution</td></tr><tr><td></td><td>using a programming language to create the solution</td></tr></tbody></table>	Stage	Description	testing	identifying of the problem and requirements	analysis	reviewing the final solution to suggest further developments	coding	making sure the program code works as expected	design	using structure diagrams, flowcharts and pseudocode to plan the solution		using a programming language to create the solution	4
Stage	Description													
testing	identifying of the problem and requirements													
analysis	reviewing the final solution to suggest further developments													
coding	making sure the program code works as expected													
design	using structure diagrams, flowcharts and pseudocode to plan the solution													
	using a programming language to create the solution													

Question	Answer	Marks
4(a)	To check that a value has been entered.	1
4(b)(i)	Type (check)	1

Question	Answer	Marks
4(b)(ii)	One mark per mark point: MP1 Condition controlled loop used MP2 Working input and re-input of value into declared variable MP3 Use of selection/loop entry/exit condition to test if input is an integer MP4 Appropriate use of error message MP5 Working complete termination of loop Example: <pre> REPEAT INPUT Number IF MOD(Number, 1) <> 0 THEN OUTPUT "Please try again" ENDIF UNTIL MOD(Number, 1) = 0 Or INPUT Number WHILE MOD(Number, 1) <> 0 (DO) OUTPUT "Please try again" INPUT Number ENDWHILE </pre>	5

PUBLISHED

Question	Answer	Marks												
5	One mark for each correct box <table border="1" data-bbox="338 284 1525 751"> <thead> <tr> <th data-bbox="338 284 674 347">Test data</th><th data-bbox="674 284 949 347">Type of test data</th><th data-bbox="949 284 1525 347">Purpose of test data</th></tr> </thead> <tbody> <tr> <td data-bbox="338 347 674 448">ABC</td><td data-bbox="674 347 949 448">Abnormal</td><td data-bbox="949 347 1525 448">to make sure that the program rejects data that is too short</td></tr> <tr> <td data-bbox="338 448 674 651">Password1 // Password22</td><td data-bbox="674 448 949 651">Boundary</td><td data-bbox="949 448 1525 651">to make sure that the program rejects data that is only just too short // to make sure that the program accepts data that is only just at the correct length</td></tr> <tr> <td data-bbox="338 651 674 751">CambridgeInternational</td><td data-bbox="674 651 949 751">Normal</td><td data-bbox="949 651 1525 751">to make sure that the program accepts data that is an appropriate length</td></tr> </tbody> </table>	Test data	Type of test data	Purpose of test data	ABC	Abnormal	to make sure that the program rejects data that is too short	Password1 // Password22	Boundary	to make sure that the program rejects data that is only just too short // to make sure that the program accepts data that is only just at the correct length	CambridgeInternational	Normal	to make sure that the program accepts data that is an appropriate length	6
Test data	Type of test data	Purpose of test data												
ABC	Abnormal	to make sure that the program rejects data that is too short												
Password1 // Password22	Boundary	to make sure that the program rejects data that is only just too short // to make sure that the program accepts data that is only just at the correct length												
CambridgeInternational	Normal	to make sure that the program accepts data that is an appropriate length												

Question	Answer	Marks
6(a)	One mark per mark point - Line 02 / DECLARE Highest : STRING should be DECLARE Highest : INTEGER - Line 05 / Highest $\leftarrow$ 1500 should be Highest $\leftarrow$ 0 - Line 09 / Total $\leftarrow$ Total + Count should be Total $\leftarrow$ Total + Numbers[Count] - Line 10 / IF Numbers[Count] > Total should be IF Numbers[Count] > Highest - Line 17 / OUTPUT "The average is ", Average / 1000 should be OUTPUT "The average is ", Total / 1000	5

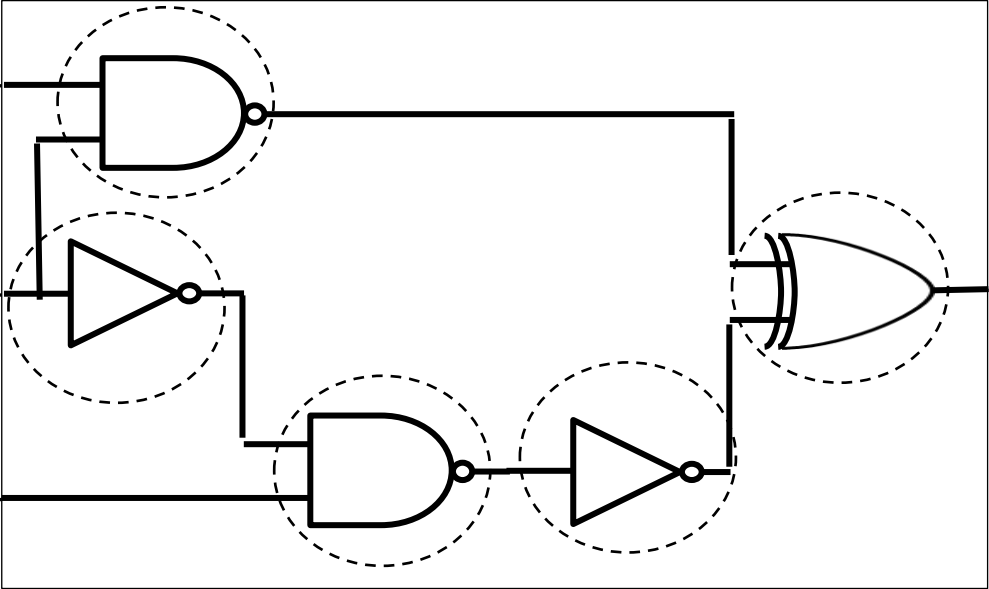
Question	Answer	Marks
6(a)	Corrected algorithm <pre> 01 DECLARE Numbers : ARRAY[1:1000] OF INTEGER 02 DECLARE Highest : INTEGER 03 DECLARE Count : INTEGER 04 DECLARE Total : INTEGER 05 Highest ← 0 06 Total ← 0 07 FOR Count ← 1 TO 1000 08 INPUT Numbers[Count] 09 Total ← Total + Numbers[Count] 10 IF Numbers[Count] > Highest 11 THEN 12 Highest ← Numbers[Count] 13 ENDIF 14 NEXT Count 15 OUTPUT "The highest number is ", Highest 16 OUTPUT "The total is ", Total 17 OUTPUT "The average is ", Total / 1000 </pre>	
6(b)	One mark per mark point MP1 Correct use of <code>ROUND</code> function to round the average MP2 ... set to 2 decimal places with <code>OUTPUT</code> command. Example: <pre>OUTPUT ROUND(Total / 1000, 2)</pre>	2
6(c)	One mark per mark point, max four MP1 Declaration of new variable for smallest value at start of algorithm/before it is used MP2 Initialisation of smallest variable to a high number / the first input MP3 New selection statement after input to compare input with current smallest number MP4 If input is smaller than current smallest variable, it should replace it MP5 Outside the loop, output the current value of the smallest variable.	4

Question	Answer	Marks																																																																	
7(a)	One mark per correct column <table><tr><th>Answer</th><th>Value1</th><th>Operator</th><th>Value2</th><th>OUTPUT</th></tr><tr><td></td><td></td><td></td><td></td><td>Continue?</td></tr><tr><td>Y</td><td>7</td><td>S</td><td>9</td><td>-2</td></tr><tr><td></td><td></td><td></td><td></td><td>Continue?</td></tr><tr><td>Y</td><td>5</td><td>M</td><td>12</td><td>60</td></tr><tr><td></td><td></td><td></td><td></td><td>Continue?</td></tr><tr><td>Y</td><td>25</td><td>D</td><td>5</td><td>5</td></tr><tr><td></td><td></td><td></td><td></td><td>Continue?</td></tr><tr><td>N</td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td></tr></table>	Answer	Value1	Operator	Value2	OUTPUT					Continue?	Y	7	S	9	-2					Continue?	Y	5	M	12	60					Continue?	Y	25	D	5	5					Continue?	N																									5
Answer	Value1	Operator	Value2	OUTPUT																																																															
				Continue?																																																															
Y	7	S	9	-2																																																															
				Continue?																																																															
Y	5	M	12	60																																																															
				Continue?																																																															
Y	25	D	5	5																																																															
				Continue?																																																															
N																																																																			
7(b)	It is a calculator.	1																																																																	

Question	Answer	Marks
7(c)(i)	One mark per mark point MP1 Include two process boxes MP2 ... after the Answer and Operator input boxes MP3 Use of UCASE in both boxes to change the input given to upper case. OR One mark per mark point MP4 Change the condition in all the (4) decision boxes MP5 ... for the variables Answer and Operator MP6 Using OR e.g. OR Answer = 'y' and OR Operator = 'a'	3
7(c)(ii)	One mark, for example: Only tests for A, S, and M; any other input is assumed to be D (for division) Divide by 0 error (if D and 0 are entered) Wrong data type e.g. character instead of number No data entered (enter without a preceding value) If Yes/yes entered the algorithm stops No input prompts.	1

Question	Answer	Marks
8(a)	One mark per correct answer Fields: 6 Records: 23	2
8(b)	One mark per correct answer Primary key field: Code Reason: It is the only field with unique contents. // It is a unique identifier	2

Question	Answer	Marks									
8(c)	One mark per mark point MP1 Correct data present – spelt correctly MP2 Correct layout – three columns, with no additional punctuation MP3 Correct order and no integrity errors Correct output <table> <tr> <td>Valencia</td> <td>Venezuela</td> <td>1,983,445</td> </tr> <tr> <td>Lima</td> <td>Peru</td> <td>11,206,000</td> </tr> <tr> <td>Buenos Aires</td> <td>Argentina</td> <td>15,490,415</td> </tr> </table>	Valencia	Venezuela	1,983,445	Lima	Peru	11,206,000	Buenos Aires	Argentina	15,490,415	3
Valencia	Venezuela	1,983,445									
Lima	Peru	11,206,000									
Buenos Aires	Argentina	15,490,415									
8(d)	One mark per mark point MP1 At least two correct fields in SELECT MP2 Remaining two correct fields in SELECT MP3 FROM MajorCity MP4 WHERE Capital = TRUE; Correct code: <pre>SELECT Code, City, Country, Continent FROM MajorCity WHERE Capital = TRUE;</pre>	4									

Question	Answer	Marks
9(a)	One mark for each correct gate, with the correct input(s) as shown. 	5

Question	Answer	Marks																																				
9(b)	Four marks for all eight correct outputs Three marks for six or seven correct outputs Two marks for four or five correct outputs One mark for two or three correct outputs <table><tr><th>A</th><th>B</th><th>C</th><th>Z</th></tr><tr><td>0</td><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>0</td><td>1</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td><td>0</td></tr></table>	A	B	C	Z	0	0	0	1	0	0	1	0	0	1	0	1	0	1	1	1	1	0	0	1	1	0	1	0	1	1	0	0	1	1	1	0	4
A	B	C	Z																																			
0	0	0	1																																			
0	0	1	0																																			
0	1	0	1																																			
0	1	1	1																																			
1	0	0	1																																			
1	0	1	0																																			
1	1	0	0																																			
1	1	1	0																																			

PUBLISHED

Question	Answer	Marks
10	Marks are available for: • AO2 (maximum 9 marks) • AO3 (maximum 6 marks) Data structures required with names as given in the scenario: Arrays or lists <u>MemberID[]</u>, <u>Name[]</u> Variables <u>NewID</u> Requirements (techniques) R1 displays menu and allows choice, then proceeds based on valid choice, (nested iteration, input, output, selection, validation). R2 checks length of string, checks contents of array for matches, stores approved data (validation, string handling (length), selection, input, storage). R3 outputs contents of arrays. Program continues until user chooses stop. (output, iteration).	15

Question	Answer	Marks
10	Example 15-mark answer in pseudocode <pre>// Array and variable declaration - not required in candidates' responses REPEAT // Display of menu choices OUTPUT "Enter 1 to input a new member, 2 to output member codes and first and last names, or 3 to stop " // Input menu choice with validation of input REPEAT INPUT Answer IF Answer < 1 OR Answer > 3 THEN OUTPUT "You must input 1, 2 or 3. Please try again " ENDIF UNTIL Answer >= 1 AND Answer <= 3 // User chooses to input new member details IF Answer = 1 THEN REPEAT // Initialisation of flag for previous use of code Used ← FALSE // User enters new code, with prompt OUTPUT "Enter a new six-character membership code " INPUT Code // Checking code is 6 characters long IF LENGTH(Code) <> 6 THEN // If code is wrong length, re-entry required OUTPUT "The code must contain six characters, please try again." ELSE // If code is correct length, it is checked with // with previous codes for uniqueness IndexCheck ← 1 WHILE MemberID[IndexCheck] <> "" AND NOT Used DO IF Code = MemberID[IndexCheck]</pre>	

PUBLISHED

Question	Answer	Marks
10	<pre> THEN // If code already used, flag changed and re-entry required Used ← TRUE OUTPUT "This code has already been used, please try again " ELSE IndexCheck ← IndexCheck + 1 ENDIF ENDWHILE // If code correct length and not previously used // it is stored, along with first and last name IF NOT Used THEN MemberID[IndexCheck] ← Code OUTPUT "Enter your first name " INPUT Name[IndexCheck, 1] OUTPUT "Enter your last name " INPUT Name[IndexCheck, 2] ENDIF ENDIF UNTIL LENGTH(Code) = 6 AND NOT Used ENDIF // User chooses to output all member details IF Answer = 2 THEN IndexOut ← 1 // All array contents output using iteration WHILE MemberID[IndexOut] <> "" OUTPUT "Membership code: ", MemberID[IndexOut] OUTPUT "First name: ", Name[IndexOut, 1] OUTPUT "Last name: ", Name[IndexOut, 2] IndexOut ← IndexOut + 1 ENDWHILE ENDIF UNTIL Answer = 3 </pre>	

Marking Instructions in italics			
AO2: Apply knowledge and understanding of the principles and concepts of computer science to a given context, including the analysis and design of computational or programming problems			
0	1–3	4–6	7–9
No creditable response.	At least one programming technique has been used. <i>Any use of selection, iteration, counting, totalling, input and output.</i>	Some programming techniques used are appropriate to the problem. <i>More than one technique seen applied to the scenario, check the list of techniques needed.</i>	The range of programming techniques used is appropriate to the problem. <i>All criteria stated for the scenario have been covered by the use of appropriate programming techniques, check the list of techniques needed.</i>
	Some data has been stored but not appropriately. <i>Any use of variables or arrays or other language dependent data structures e.g. Python lists.</i>	Some of the data structures chosen are appropriate and store some of the data required. <i>More than one data structure used to store data required by the scenario.</i>	The data structures chosen are appropriate and store all the data required. <i>The data structures used store all the data required by the scenario.</i>

Marking Instructions in italics			
AO3: Provide solutions to problems by: evaluating computer systems making reasoned judgements presenting conclusions			
0	1–2	3–4	5–6
No creditable response.	Program seen without relevant comments.	Program seen with some relevant comment(s).	The program has been fully commented
	Some identifier names used are appropriate. <i>Some of the data structures used have meaningful names.</i>	The majority of identifiers used are appropriately named. <i>Most of the data structures used have meaningful names.</i>	Suitable identifiers with names meaningful to their purpose have been used throughout. <i>All of the data structures used have meaningful names.</i>
	The solution is illogical.	The solution contains parts that may be illogical.	The program is in a logical order.
	The solution is inaccurate in many places. <i>Solution contains few lines of code with errors that attempt to perform a task given in the scenario.</i>	The solution contains parts that are inaccurate. <i>Solution contains lines of code with some errors that logically perform tasks given in the scenario. Ignore minor syntax errors.</i>	The solution is accurate. <i>Solution logically performs all the tasks given in the scenario. Ignore minor syntax errors.</i>
	The solution attempts at least one of the requirements. <i>Solution contains lines of code that attempt at least one task given in the scenario.</i>	The solution attempts to meet most of the requirements. <i>Solution contains lines of code that attempt most tasks given in the scenario.</i>	The solution meets all the requirements given in the question. <i>Solution performs all the tasks given in the scenario.</i>