GradeMax

CAMBRIDGE
International Education

Cambridge IGCSE™

COMPUTER SCIENCE **0478/22**

Paper 2 Algorithms, Programming and Logic **October/November 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 October/November 2025 series for most Cambridge IGCSE, Cambridge International A and AS Level components, and some Cambridge O Level components.

© Cambridge University Press & Assessment 2025

GradeMax

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.

© Cambridge University Press & Assessment 2025

GradeMax

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.

© Cambridge University Press & Assessment 2025

GradeMax

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

© Cambridge University Press & Assessment 2025

GradeMax

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

© Cambridge University Press & Assessment 2025

GradeMax

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.

© Cambridge University Press & Assessment 2025

GradeMax

Question	Answer	Marks
1	C	**1**

Question	Answer	Marks
2(a)	**One** mark per mark point, **max five** • Line 01 / `DECLARE Contacts : ARRAY[1:500, 1:4] OF REAL` should be `DECLARE Contacts : ARRAY[1:500, 1:4] OF STRING` • Line 12 / `OUTPUT FirstName` should be `INPUT FirstName` • Line 17 / `OUTPUT Contacts[Row, 1]` should be `OUTPUT Contacts[Row, Column]` • Line 23 / `NEXT Stop OR Row > 500` should be `UNTIL Stop OR Row > 500` • Line 25 / 26 / Answer variable not declared should be `DECLARE Answer : CHAR` before line 25 • Line 28 / `Continue ← TRUE` should be `Continue ← FALSE`	**5**

© Cambridge University Press & Assessment 2025

Question	Answer	Marks

2(b)(i)	**One** mark per mark point, **max four**
	
	MP1 Appropriate parameter in `PROCEDURE` line
	MP2 Correct index outer loop termination value; must use `Number` as stated in question
	MP3 Correct array reference in `INPUT`
	MP4 Correct termination of both loops.
	
	Example:
	
	`PROCEDURE NewData(Number : INTEGER)`
	`  FOR Row ← 1 TO Number`
	`  FOR Column ← 1 TO 4`
	`  INPUT Contacts[Row, Column]`
	`  NEXT Column`
	`  NEXT Row`
	`ENDPROCEDURE`
		4

2(b)(ii)	**One** mark per mark point, **max two**
	
	MP1 Correct input statement
	MP2 Correct call statement for `NewData`
	
	Example:
	
	`INPUT MyNumber`
	`CALL NewData(MyNumber)`
		2

2(c)(i)	**One** mark per mark point, **max two** MP1 There may be more than one person with the given first name // The name may not exist/found in the array MP2 The search will stop at the first occurrence and **output the data for the wrong person** MP3 If the name is not present there could be no output / the loop will continue.	2

© Cambridge University Press & Assessment 2025

GradeMax

Question	Answer	Marks
2(c)(ii)	**One** mark per mark point, **max two** MP1 Introduce additional search criteria into the algorithm **to make the search more specific to each person in the array** MP2 … such as the last name to go with the first name // … such as some form of ID number / date of birth / contact number / post code MP3 The algorithm should be able to search the whole array to find all available matches MP4 … by removing the stop variable MP5 Use selection to confirm the name has not been found in the whole array MP6 If the name is not present, output an appropriate message.	**2**

Question	Answer	Marks
3	**One** mark for each correct box (underlined), **max six**	**6**

Test data	Type of test data	Purpose of test data
0	normal	to make sure that the program **accepts** data that is in range
100 // –100	*boundary*	to make sure that the program **rejects** data that is on the wrong side of the boundary
300	abnormal	*to make sure that the program rejects data that is outside acceptable limits*

© Cambridge University Press & Assessment 2025

GradeMax

Question	Answer	Marks
4(a)	To check that the data entered has the required number of characters.	1

4(b)	**One** mark per mark point, **max six**
	
	MP1 Initialisation of variables – start location for checking and flag, if used
	MP2 Input of email address into appropriate variable
	MP3 Working condition-controlled loop present
	MP4 Correct use of `SUBSTRING`
	MP5 Use of condition for checking value extracted from `SUBSTRING` / `'@'`
	MP6 Appropriate action if `'@'` found – loop termination
	MP7 Appropriate action if `'@'` NOT found – check the next character **and** loop termination if no more characters
	MP8 Output that **only** shows when the email address is valid or invalid. **Both** cases required.
	
	Example:
	
	`// Variable declarations are not required`
	`INPUT EmailAddress`
	
	`Start ← 1`
	`Valid ← FALSE`
	`REPEAT`
	`  IF SUBSTRING(EmailAddress, Start, 1) = '@'`
	`  THEN`
	`  Valid ← TRUE`
	`  ELSE`
	`  Start ← Start + 1`
	`  ENDIF`
	`UNTIL Start > LENGTH(EmailAddress) OR Valid`
	`IF Valid`
	`  THEN`
	`  OUTPUT EmailAddress, " is valid"`
	`  ELSE`
	`  OUTPUT EmailAddress, " is NOT valid"`
	`ENDIF`
		6

© Cambridge University Press & Assessment 2025

Question	Answer	Marks
5(a)	**One** mark per mark point, **max six** MP1 Correct `Code` column MP2 Correct `Store` column MP3 Correct `Count` column MP4 First four `CodeStore` columns left unchanged MP5 Correct `CodeStore[5]` and `CodeStore[6]` columns MP6 Correct OUTPUT column	**6**

			CodeStore[]						
Code	**Store**	**Count**	**[1]**	**[2]**	**[3]**	**[4]**	**[5]**	**[6]**	**OUTPUT**
			aBcd	**1532**	**Face**	**63q8**			
7686	FALSE	1							
		2							
		3							
		4							
	TRUE	5					7686		
		6							
Face	FALSE	1							
		2							

© Cambridge University Press & Assessment 2025

Question	Answer	Marks

5(a)

			CodeStore[]						
Code	**Store**	**Count**	**[1]**	**[2]**	**[3]**	**[4]**	**[5]**	**[6]**	**OUTPUT**
		3							Code used before
Speed									Length must be 4
432U	FALSE	1							
		2							
		3							
		4							
		5							
	TRUE	6						432U	
		7							CodeStore full

© Cambridge University Press & Assessment 2025

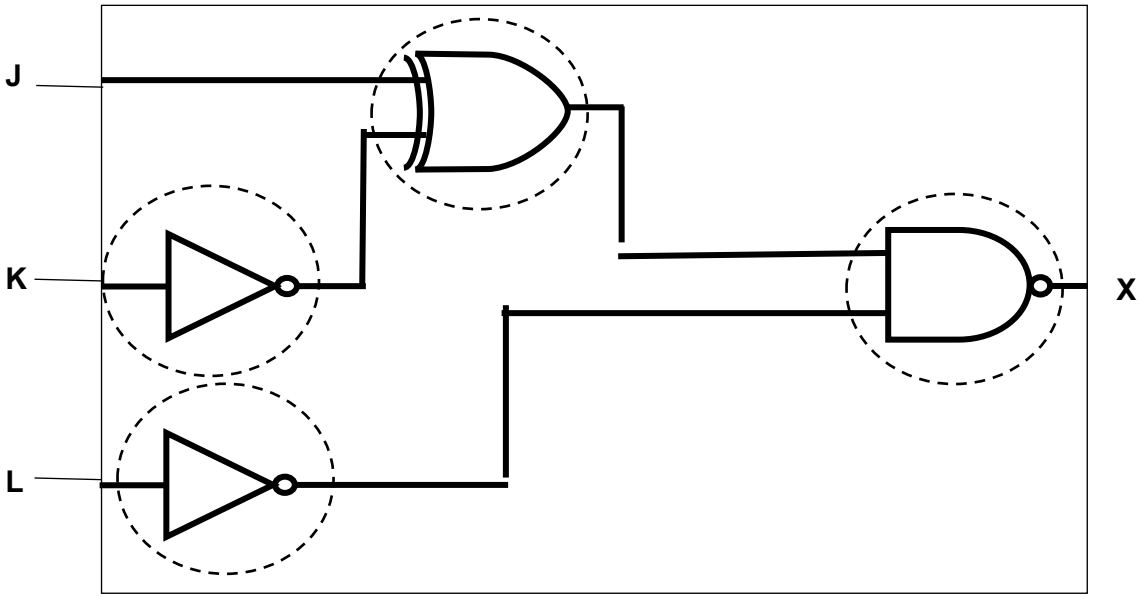
GradeMax

Question	Answer	Marks
5(b)	**One** mark per mark point, **max three** MP1 Checks that the length of the code input is 4 characters. MP2 Checks the code against the array contents **to see if it has been used before** (in the last 6 codes) MP3 If the code passes the tests, it is stored in the array MP4 Stores the code in the lowest available index MP5 Checks that the code store isn't full before storing a new code // Outputs a code store is full message if appropriate // Algorithm continues until the code store is full.	**3**

5(c)	**One** mark per mark point, **max four**
	
	MP1 Declaration of array using correct name and data type
	MP2 Declaration of loop counter
	MP3 Use of appropriate loop for six iterations (allow any type of loop)
	MP4 Assignment of all array elements to null string (using loop counter as array index)
	
	Example:
	
	`DECLARE CodeStore : ARRAY[1:6] OF STRING`
	`DECLARE Index : INTEGER`
	`FOR Index ← 1 TO 6`
	`  CodeStore[Index] ← ""`
	`NEXT Index`
		4

© Cambridge University Press & Assessment 2025

GradeMax

Question	Answer	Marks
6(a)	**One** mark for each correct gate, with the correct input(s) as shown. 	**4**

© Cambridge University Press & Assessment 2025

GradeMax

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>6(b)</td><td>Four marks for 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>J</th><th>K</th><th>L</th><th>X</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</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>1</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>1</td></tr>
</table></td><td>4</td></tr>
</table>

© Cambridge University Press & Assessment 2025

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>7(a)</td><td>It is used to uniquely identify records</td><td>1</td></tr>
<tr><td>7(b)</td><td>Two marks for all four fields correctly identified
One mark for two or three fields correctly identified

<table>
<tr><th>Data type</th><th>Field</th></tr>
<tr><td>Boolean</td><td><code>CoastalHabitat</code></td></tr>
<tr><td>Integer</td><td><code>Lifespan</code></td></tr>
<tr><td>Real</td><td><code>Length</code> // <code>Wingspan</code></td></tr>
<tr><td>Text</td><td><code>Species</code> // <code>Family</code></td></tr>
</table></td><td>2</td></tr>
<tr><td>7(c)</td><td>One mark per mark point, max three

MP1 Correct data – spelt correctly
MP2 Correct layout – three columns, with no additional punctuation
MP3 Correct order (vertical and horizontal)

Correct output:

<pre>Bewick’s swan 120.2 190.5
Whooper swan 150.0 230.0
Mute swan 150.0 220.2</pre></td><td>3</td></tr>
<tr><td>7(d)</td><td>One mark per correct line, max four

<pre>SELECT Species, Family, Lifespan
FROM WATERFOWL
WHERE Lifespan >= 10 ORDER BY Species;</pre></td><td>4</td></tr>
</table>

GradeMax

Question	Answer	Marks

8	Use the tables for AO2 and AO3 below to award a mark in a suitable band using a best fit approach, then add up the total.
	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 `RandomNumber[]`, `CountedNumber[]`
	
	Requirements (techniques)
	
	R1 arrays and variables declared, then 100 000 random integers between 1 and 10, inclusive, generated and stored in an array, (array and variable declarations, iteration, random number generation, rounding and storage).
	R2 frequency of each random integer counted and results stored in array, then results sorted into descending order (initialisation, selection, counting, sort, nested iteration, storage).
	R3 calculation of the chance of each random integer occurring rounded to 4 d.p. and results output (iteration, rounding and output).
	
	Example 15-mark answer in pseudocode
	
	`// array and variable declaration`
	`DECLARE RandomNumber : ARRAY[1:100000] OF INTEGER`
	`DECLARE CountedNumber : ARRAY[1:10, 1:2] OF INTEGER`
	`DECLARE Index1 : INTEGER`
	`DECLARE Index2 : INTEGER`
	`DECLARE Swap : BOOLEAN`
	`DECLARE Temp1 : INTEGER`
	`DECLARE Temp2 : INTEGER`
	`// generate 100000 random integers between 1 and 10, inclusive,`
	`// and store them in an array`
	`FOR Index1 ← 1 TO 100000`
	`  RandomNumber[Index1] ← ROUND(RANDOM() * 9, 0) + 1`
	`NEXT Index1`
		15

© Cambridge University Press & Assessment 2025

Question	Answer	Marks

8	`// initialising the CountedNumber array`
	`FOR Index1 ← 1 TO 10`
	`  CountedNumber[Index1, 1] ← Index1`
	`  CountedNumber[Index1, 2] ← 0`
	`NEXT Index1`
	`// counting the frequency of each of the random integers`
	`// and storing the frequencies in an array`
	`FOR Index1 ← 1 TO 10`
	`  FOR Index2 ← 1 TO 100000`
	`  IF RandomNumber[Index2] = CountedNumber[Index1, 1]`
	`  THEN`
	`  CountedNumber[Index1, 2] ← CountedNumber[Index1, 2] + 1`
	`  ENDIF`
	`  NEXT Index2`
	`NEXT Index1`
	`// performing sort on frequencies, so the possible random numbers`
	`// are stored in descending order of frequency`
	`Swap ← TRUE`
	`WHILE Swap DO`
	`  // sort is terminated when no swaps take place during a pass`
	`  Swap ← FALSE`
	`  FOR Index1 ← 1 TO 9`
	`  IF CountedNumber[Index1, 2] < CountedNumber[Index1 + 1, 2]`
	`  THEN`
	`  Temp1 ← CountedNumber[Index1, 1]`
	`  Temp2 ← CountedNumber[Index1, 2]`
	`  CountedNumber[Index1, 1] ← CountedNumber[Index1 + 1, 1]`
	`  CountedNumber[Index1, 2] ← CountedNumber[Index1 + 1, 2]`
	`  CountedNumber[Index1 + 1, 1] ← Temp1`
	`  CountedNumber[Index1 + 1, 2] ← Temp2`
	`  Swap ← TRUE`
	`  ENDIF`
	`  NEXT Index1`
	`ENDWHILE`	

© Cambridge University Press & Assessment 2025

Question	Answer	Marks

8	`// outputting the results, with chances for each random number`
	`// calculated to 4 d.p.`
	`// results in descending order of frequency`
	`FOR Index1 ← 1 TO 10`
	`  OUTPUT "The chance of ", CountedNumber[Index1, 1], " occurring is: "`
	`  OUTPUT ROUND(CountedNumber[Index1, 2]/100000, 4)`
	`NEXT Index1`
	
	Alternative answer for first 20 lines after declarations, where `RandomNumber[]` is populated and frequency counted in the same loop.
	`// initialising the CountedNumber array`
	`FOR Index1 ← 1 TO 10`
	`  CountedNumber[Index1, 1] ← Index1`
	`  CountedNumber[Index1, 2] ← 0`
	`NEXT Index1`
	`// generate random integers in an array,`
	`// counting and storing the frequencies in an array`
	`FOR Index1 ← 1 TO 100000`
	`  Temp1 ← ROUND(RANDOM() * 9, 0) + 1`
	`  RandomNumber[Index1] ← Temp1`
	`  CountedNumber[Temp1, 2] ← CountedNumber[Temp1, 2] + 1`
	`NEXT Index1`
	`// performing sort on frequencies, so the possible random numbers`	

© Cambridge University Press & Assessment 2025

GradeMax

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. *Any use of selection, iteration, counting, totalling, input and output.*	Some programming techniques used are appropriate to the problem. *More than one technique seen applied to the scenario, check the list of techniques needed.*	The range of programming techniques used is appropriate to the problem. *All criteria stated for the scenario have been covered by the use of appropriate programming techniques, check the list of techniques needed.*
	Some data has been stored but not appropriately. *Any **use** of variables or arrays or other language dependent data structures e.g. Python lists.*	Some of the data structures chosen are appropriate and store some of the data required. *More than one data structure **used** to store data required by the scenario.*	The data structures chosen are appropriate and store all the data required. *The data structures **used** store all the data required by the scenario.*

© Cambridge University Press & Assessment 2025

GradeMax

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. *Some of the data structures used have meaningful names.*	The majority of identifiers used are appropriately named. *Most of the data structures used have meaningful names.*	Suitable identifiers with names meaningful to their purpose have been used throughout. *All of the data structures used have meaningful names.*
	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. *Solution contains few lines of code with errors that attempt to perform a task given in the scenario.*	The solution contains parts that are inaccurate. *Solution contains lines of code with some errors that logically perform tasks given in the scenario. Ignore minor syntax errors.*	The solution is accurate. *Solution logically performs all the tasks given in the scenario. Ignore minor syntax errors.*
	The solution attempts at least one of the requirements. *Solution contains lines of code that attempt at least one task given in the scenario.*	The solution attempts to meet most of the requirements. *Solution contains lines of code that attempt most tasks given in the scenario.*	The solution meets all the requirements given in the question. *Solution performs all the tasks given in the scenario.*

© Cambridge University Press & Assessment 2025