CAMBRIDGE
International Education

Cambridge IGCSE™

COMPUTER SCIENCE **0478/23**

Paper 2 Algorithms, Programming and Logic **October/November 2024**

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 2024 series for most Cambridge IGCSE, Cambridge International A and AS Level components, and some Cambridge O Level components.

This document consists of **14** printed pages.

© Cambridge University Press & Assessment 2024

[Turn over

Generic Marking Principles

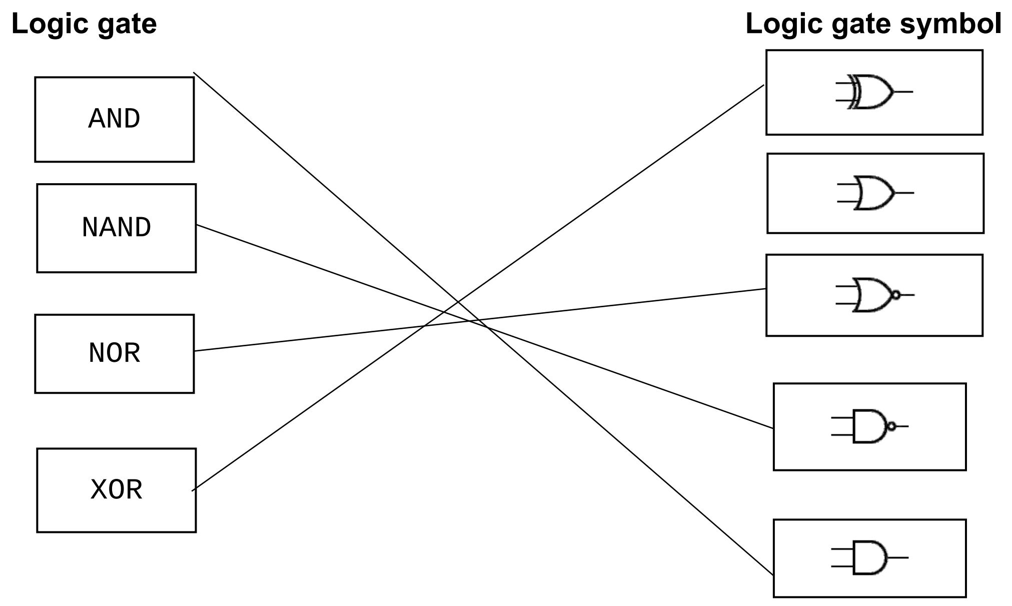
These general marking principles must be applied by all examiners when marking candidate answers. They should be applied alongside the specific content of the mark scheme or generic level descriptions for a question. Each question paper and mark scheme will also comply with these marking principles.

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.

Question	Answer	Marks
1	**C**	**1**

Question	Answer	Marks
2	**B**	**1**

Question	Answer	Marks
3	**One** mark for each correct line 	**4**

Question	Answer	Marks
4	**One** mark for each correct term Database tables consist of **rows/columns/fields/records** and **columns/rows/fields/records** Rows are **records**. **columns** are fields. Structured Query Language (SQL) **scripts** are used to query data. A **primary key** uniquely identifies a record.	**6**

Question	Answer	Marks
5	**One** mark for stage, **one** mark for matching description and **one** mark for matching expansion (**max six**) For example: • design (1) construction of a solution (1) using standard methods e.g. flowcharts (1) • coding (1) program is written (1) iterative testing takes place (1) • testing (1) program is tested for errors (1) program is tested that it meets its requirements (1)	**6**

Question	Answer	Marks
6(a)	**One** mark for each point • `06 C ← 1` • `08 W ← W + A[C]` • `11 X ← W / (C – 1) // ROUND(W / (C – 1),0)`	**3**

6(b)	**One** mark for outputting `X`
	One mark for outputting `C - 1`
	One mark for suitable messages
	
	Example:
	
	`12 OUTPUT "Number of values stored in the array is ", C - 1`
	`13 OUTPUT "Average of non-zero elements in the array is ", X`
		3

Question	Answer	Marks

6(c)	**One** mark for meaningful identifier for the array
	`A` **`Values`**
	
	Two marks for 3 meaningful identifiers for the variables
	or
	One mark for 1 to 2 meaningful identifiers for the variables
	
	`C` **`Index`**
	`X` **`Average`**
	`W` **`Total`**
		3

Question	Answer	Marks
7	**One** mark for test data type, **one** mark for matching example and one mark for matching outcome (**Max nine**) • normal (1) e.g. 25 (1) accepted (1) • extreme (1) 1/100 (1) accepted (1) • abnormal (1) e.g. 125 (1) rejected (1) • boundary (1) 1 and 0 // 100 and 101 (1) first value is accepted, and second value rejected (1)	**9**

Question	Answer	Marks
8	4 marks for 8 correct outputs 3 marks for 6/7 correct outputs 2 marks for 4/5 correct outputs 1 mark for 2/3 correct outputs	**4**

A	T	H	W
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Question	Answer	Marks
9	**One** mark for each error identified and correction • first decision box flow line labels `Yes` should be `No` and vice versa • second decision box `OR` should be `AND` • Flow line from A to B should be should `A ← B` • Flowchart symbol for `A ← C` should be a process box / rectangle	**4**

Question	Answer	Marks
10(a)	MYWORD	**6**

Password	**Accept**	**Index**	**Found**	**OUTPUT**
MYWORD	TRUE			(Please enter password)
	FALSE			
	FALSE			
		1	FALSE	
	FALSE			Rejected

M!word

Password	**Accept**	**Index**	**Found**	**OUTPUT**
M!word	TRUE			(Please enter password)
	FALSE			
		1	FALSE	
	FALSE			Rejected

Question	Answer	Marks
10(a)	My!Hidden **Password** | **Accept** | **Index** | **Found** | **OUTPUT** `My!Hidden` | `TRUE` | | | `(Please enter password)` | | `1` | `FALSE` | | | `2` | `FALSE` | | | `3` | `TRUE` | | | `4` | | | | | | `Accepted` For each trace table, **one mark** Password and output, **one mark** `Accept`, `Index` and `Found` columns	
10(b)	**One** mark for each correct point • maximum length 20 characters **and** minimum length 8 characters • cannot be all upper-case letters or all lower-case letters // must contain upper-case and lower-case letters • must contain an !	**3**

Password	Accept	Index	Found	OUTPUT
My!Hidden	TRUE			(Please enter password)
		1	FALSE	
		2	FALSE	
		3	TRUE	
		4		
				Accepted

Question	Answer	Marks

11(a)	**One** mark for each line (**max two**)
	One mark for order of lines
	
	`MT12 Pebbles large`
	`MT06 Cobbles`
		3

Question	Answer	Marks

11(b)(i)	**One** mark for correct `SELECT` and `FROM` clauses, one mark for correct `WHERE` clause
	
	`SELECT Name`
	`FROM BuildStock`
	`WHERE NOT InStock` // `InStock = FALSE` / `No` /`"No"` // `WHERE NumberBags = 0;`
		2

11(b)(ii)	**Two** marks for either point • `WHERE` clause can check the `InStock` field (1) for `FALSE` / `No` (1) or • `WHERE` clause can check the `NumberBags` field (1) for `0` (1)	**2**

Question	Answer	Marks

12	• AO2 (maximum 9 marks)
	• AO3 (maximum 6 marks)
	
	Data Structures required names shown underlined must match those given in the scenario
	2D Array or list `PickerName[]`, `PickedWeight[]`, `PickerCertificate[]`
	Variables
	
	Requirements (techniques)
	R1 Input and validate the weights (input and iteration)
	R2 sort the `PickerName[]`, and `PickedWeight[]`arrays in descending order of weight (nested iteration and sorting)
	R3 Output top two members and the weights. Storing the members names who will receive a certificate and outputting the number of certificates (iteration, selection, assignment, counting and output with appropriate messages)
	
	Example 15-mark answer in pseudocode
	
	`CONSTANT GroupSize = 200  // setting the number of members in the club`
	`DECLARE Position : Array[1:2} OF STRING`
	
	`Position[1] ← "Best in Group"`
	`Position[2] ← "Second best in Group"`
	
	`FOR Index ← 1 TO GroupSize`
	`    REPEAT`
	`        PRINT "Please enter the weight for ", PickerName[Index],`
	`          INPUT PickedWeight[Index]`
	`    UNTIL PickedWeight[Index] > 0 AND PickedWeight[Index] < 15`
	`NEXT Index`
	`Last ← GroupSize`
	`REPEAT`
	`    Swap ← FALSE`
		15

Question	Answer	Marks

12	
	```
	    FOR Index ← 1 TO GroupSize -1
	        IF PickedWeight[Index] < PickedWeight[Index + 1]
	          THEN
	            TempWeight ← PickedWeight[Index]
	            PickedWeight[Index] ← PickedWeight[Index + 1]
	            PickedWeight [Index + 1] ← TempWeight
	            TempName ← PickerName[Index]
	            PickerName[Index] ← PickerName[Index + 1]
	            PickerName[Index + 1] ← TempName
	            Swap ← TRUE
	        ENDIF
	    NEXT Index
	    Last ← Last - 1
	UNTIL NOT Swap or Last = 1
	FOR Index ← 1 TO 2
	    OUTPUT Position[Index], PickerName[Index], " with a weight of ", PickedWeight[Index]
	NEXT Index
	Index ← 1
	Count ← 0
	WHILE PickedWeight[Index] > 3 DO // count number of certificates required and store names
	    Count ← Count + 1
	    PickerCertificate[Count] ← PickerName[Index]
	    Index ← Index + 1
	ENDWHILE
	OUTPUT "Number of certificates to be printed is ", Count
	```

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 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 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.*

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 meets most of the requirements. *Solution contains lines of code that perform 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.*