CAMBRIDGE
International Education

MEM

Cambridge IGCSE™

COMPUTER SCIENCE **0478/22**

Paper 2 Algorithms, Programming and Logic **February/March 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 February/March 2024 series for most Cambridge IGCSE, Cambridge International A and AS Level components, and some Cambridge O Level components.

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.

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	**B**	**1**

Question	Answer	Marks
2(a)	**One** mark for each correct line from the test data type the description **Test data type** **Description** abnormal → a value that is rejected boundary → a value that is the highest or lowest value to be accepted and the corresponding lowest or highest value to be rejected extreme → a value that is the highest or lowest value to be accepted normal → a value that is accepted a value that is the highest or lowest value to be rejected a value that is rejected	**4**
2(b)	**One** mark for each point: • Abnormal for example 31 • Boundary 4, 5 // 10, 11 • Extreme 5 // 10 • Normal for example 6	**4**

Question	Answer	Marks

3(a)	**One** mark for each point **max four**.
	• input value, outside loop
	• correct use of loop
	• checking value input against contents of array …
	• ... appropriate action
	• correct outputs
	
	Example:
	
	`INPUT MyNumber`
	`Location ← 0`
	`FOR Index ← 1 TO 50`
	`  IF Values[Index] = MyNumber`
	`  THEN`
	`  Location ← Index`
	`  ENDIF`
	`NEXT Index`
	`IF Location = 0`
	`  THEN`
	`  OUTPUT "Not found"`
	`  ELSE`
	`  OUTPUT Location`
	`ENDIF`
		4

Question	Answer	Marks

3(b)	**One** mark for each point **max four**.
	• use of inner and outer loop
	• correct use of loops
	• checking adjacent values in array …
	• ... swap if required
	• correct stopping condition
	
	`Last ← 50`
	`Repeat`
	`  Swap ← FALSE`
	`  FOR Index ← 1 TO Last - 1`
	`  IF Values[Index] > Values[Index + 1]`
	`  THEN`
	`  Temp ← Values[Index]`
	`  Values[Index] ← Values[Index + 1]`
	`  Values[Index + 1] ← Temp`
	`  Swap ← TRUE`
	`  ENDIF`
	`  NEXT`
	`  Last ← Last - 1`
	`UNTIL NOT Swap or Last = 1`
		4

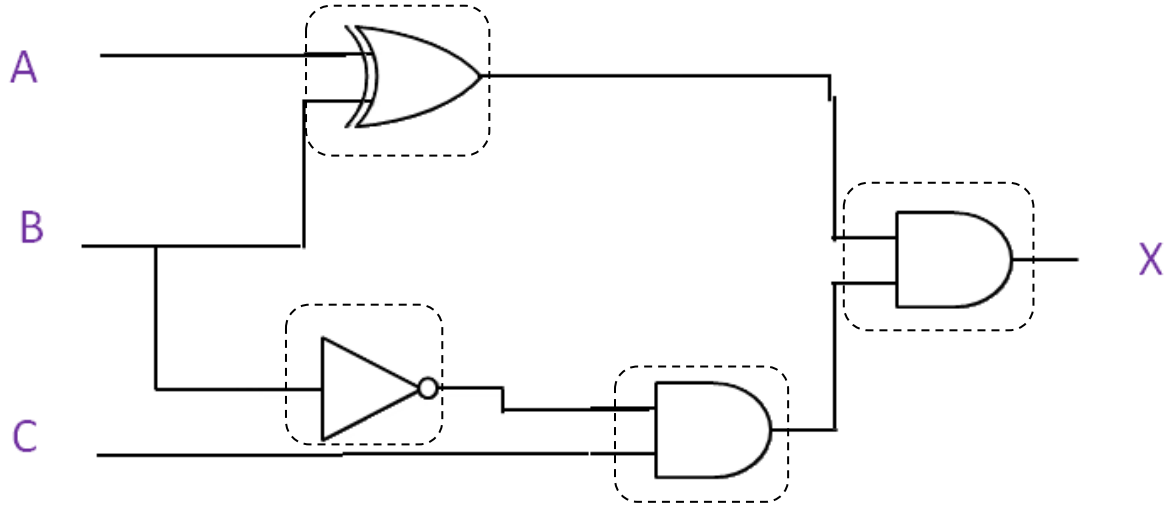
Question	Answer	Marks
4	**One** mark for each point **max three**. • Integer • real • char • string • Boolean	**3**

Question	Answer	Marks
5(a)	**One** mark for each error identified and correction • Line `05 OUTPUT` should be `INPUT` • Line `06 OR` should be `AND` • Line `10 NEXT` should be `ENDIF`	**3**

5(b)(i)	**One** mark for checking for `< 0` or `>= 0`
	One mark for checking of both inputs
	One mark for correct repetition of both inputs e.g. use of `REPEAT WHILE` or using existing loop, in separate loops or both in a single loop
	
	Example:
	
	`REPEAT`
	`  OUTPUT "Enter cost price "`
	`  INPUT Cost`
	`UNTIL Cost >= 0`
	`REPEAT`
	`  OUTPUT "Enter selling price "`
	`  INPUT Sell`
	`UNTIL Sell >= 0`
	
	or
	
	`OUTPUT "Enter cost price "`
	`INPUT Cost`
	`WHILE Cost < 0`
	`  OUTPUT "Enter cost price "`
	`  INPUT Cost`
	`ENDWHILE`
	`OUTPUT "Enter selling price "`
	`INPUT Sell`
	`WHILE Sell < 0`
	`  OUTPUT "Enter selling price "`
	`  INPUT Sell`
	`ENDWHILE`
		3

Question	Answer	Marks
5(b)(ii)	**One** mark for identifying validation check and **one** mark for accompanying description **max four** • presence check (1) to check that values have been input (1) • type check (1) to check for numerical values (1)	**4**

Question	Answer	Marks
6	**One** mark for identifying a type of iteration, **one** mark for accompanying description **max four** • count controlled (1) number of iterations is pre-determined (1) • pre-condition (1) checks condition at start of loop // loop may not iterate (1) • post-condition (1) checks condition at end of loop // loop always iterates at least once (1)	**4**

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

Question	Answer	Marks
7(b)		**4**

A	B	C	X
0	0	0	**0**
0	0	1	**0**
0	1	0	**0**
0	1	1	**0**
1	0	0	**0**
1	0	1	**1**
1	1	0	**0**
1	1	1	**0**

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

Question	Answer	Marks
8	**One** mark for each column `Total`, `Average` and `OUTPUT` **One** mark for columns `NumberGroups` and `GroupSize`	**4**

NumberGroups	Total	GroupSize	Average	OUTPUT
0	0			
1	7	7		
2	17	10		
3	19	2		
4	27	8		
5	30	3		
6	39	9		
		0	6	Average group size 6

Question	Answer	Marks
9(a)(i)	`StorageID`	**1**
9(a)(ii)	It is a unique identifier	**1**

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>9(b)</td><td>One mark for every two correct data types

<table>
<tr><th>Field</th><th>Data type</th></tr>
<tr><td>SizeMetres</td><td>Real/Integer</td></tr>
<tr><td>Position</td><td>Char/Integer/Text/Alphanumeric</td></tr>
<tr><td>Hoist</td><td>Boolean/Text/Alphanumeric</td></tr>
<tr><td>StorageID</td><td>Text/Alphanumeric</td></tr>
</table></td><td>2</td></tr>
<tr><td>9(c)</td><td>One mark if two correct or two marks if completely correct
<pre>SELECT StorageID, PriceMonth, SizeMetres</pre>
One mark each point max two
<pre>FROM StorageUnits
WHERE Hoist = TRUE;</pre></td><td>4</td></tr>
</table>

Question	Answer	Marks
10	**One** mark for each technique **One** mark for a matching description **max six** • use comments … • … to explain the purpose of each section of code • …for example, logic / syntax • use meaningful identifier names to … • … clearly identify the purpose • … of variables, constants, arrays, procedures etc • … by example • use procedures and functions … • … to avoid repeated code • … simplify logic • use indentation and white space … • … to make the program readable	**6**

Question	Answer	Marks
11	Check if each requirement listed below has been met. Requirements may be met using a suitable built-in function from the programming language used (Python, VB.NET or Java). On the script, add seen if the requirement has been met, NE if a partial attempt, or a cross if no attempt. 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 `StudentName`, `ScreenTime` Variable `ClassSize` could be constant **Requirements (techniques)** **R1** allows a student to enter their weekly screen time and calculates the total number of minutes of screen time for each student in the week (input, iteration and totalling) **R2** counts the number of days with more than 300 minutes screen time each day and calculates the average week's screen time for the whole class (selection, counting, iteration, calculating average) **R3** finds the student with the lowest weekly minutes. Outputs for each student: name, total week's screen time in hours and minutes, number of days with more than 300 minutes screen time, outputs the average weeks screen time for the whole class and the name of the student with the lowest number of minutes (finding minimum value, output)	**15**

Question	Answer	Marks
11	***Example 15-mark answer in pseudocode***	

```
WeekLength ← 5
LowestMinutes ← 1000
ClassTotal← 0
FOR StudentCounter ← 1 to ClassSize // loop for each student
    Total ← 0
    DaysOver300 ← 0
    FOR DayCounter ← 1 to WeekLength // loop for each day
        REPEAT
            OUTPUT "Please enter number of minutes for day ", DayCounter
            INPUT Minutes
        UNTIL Minutes >= 0
        ScreenTime[StudentCounter, DayCounter] ← Minutes
        Total ← Total + Minutes
        IF Minutes > 300
          THEN
            DaysOver300 ← DaysOver300 + 1
        ENDIF
        IF Minutes < LowestMinutes
          THEN
            LowestMinutes ← Minutes
            LowestIndex ← StudentCounter
        ENDIF
    NEXT DayCounter
    OUTPUT StudentName[StudentCounter]
    OUTPUT "Screen time ", DIV(Total, 60), " hours ", MOD(Total, 60), " minutes "
    OUTPUT "Days with more than 300 minutes screen time ", DaysOver300
    ClassTotal ← ClassTotal + Total
NEXT StudentCounter

OUTPUT "Average weekly screen time for class ", ClassTotal / ClassSize, " minutes "
OUTPUT "Lowest weekly time ", StudentNames[LowestIndex]
```

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.*

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.*