Cambridge Assessment International Education

Cambridge IGCSE™

COMPUTER SCIENCE **0478/22**

Paper 2 Algorithms, Programming and Logic **February/March 2023**

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 2023 series for most Cambridge IGCSE™, Cambridge International A and AS Level components and some Cambridge O Level components.

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 descriptors 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.

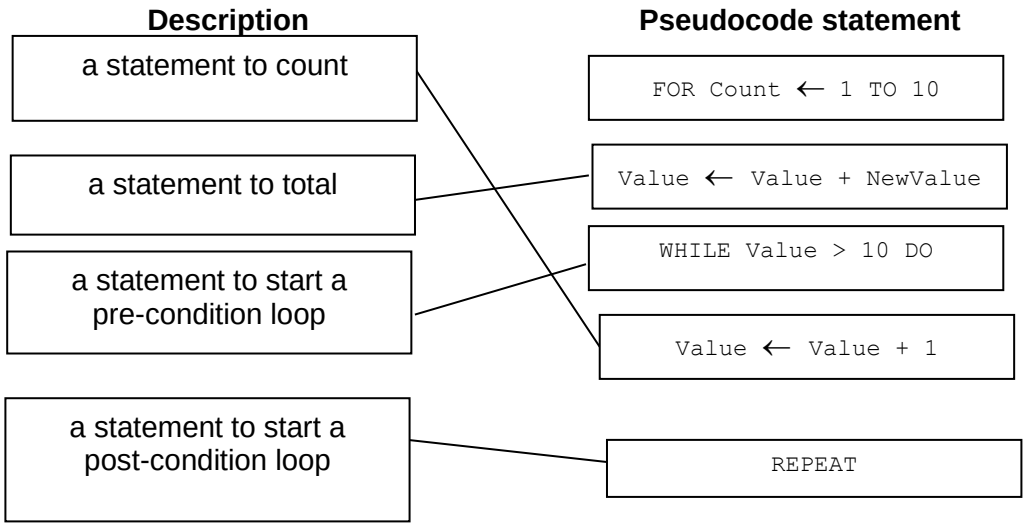
GENERIC MARKING PRINCIPLE 5: Marks should be awarded using the full range of marks defined in the mark scheme for the question (however; the use of the full mark range may be limited according to the quality of the candidate responses seen).
GENERIC MARKING PRINCIPLE 6: Marks awarded are based solely 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	**C**	**1**

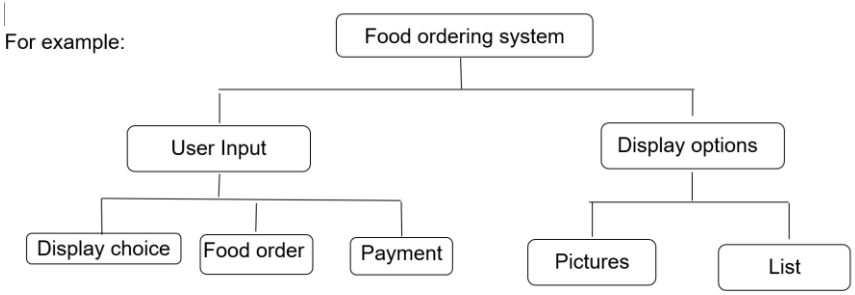
Question	Answer	Marks
2(a)	**One** mark for each **single** correct line from a description to statement 	**4**

Question	Answer	Marks

2(b)	**One** mark for each point:
	• Initialisation of total to zero before loop
	• appropriate loop controls
	• totalling statement inside the loop, must use array `Number[]` with an index
	• calculation of average outside loop
	• output of average outside loop
	
	Example:
	`Total ← 0`
	`FOR Count ← 1 TO 50`
	`  Total ← Total + Number[Count]`
	`NEXT Count`
	`Average ← Total / 50`
	`OUTPUT "The average is ", Average`
		5

Question	Answer	Marks
3	**One** mark for each point **max two**. • check that the program works as expected • check for **logic/runtime** errors • check that the program **rejects any invalid** data that is input • check that the program **only accepts** reasonable data **One** mark for example Normal // erroneous // abnormal // extreme // boundary	**3**

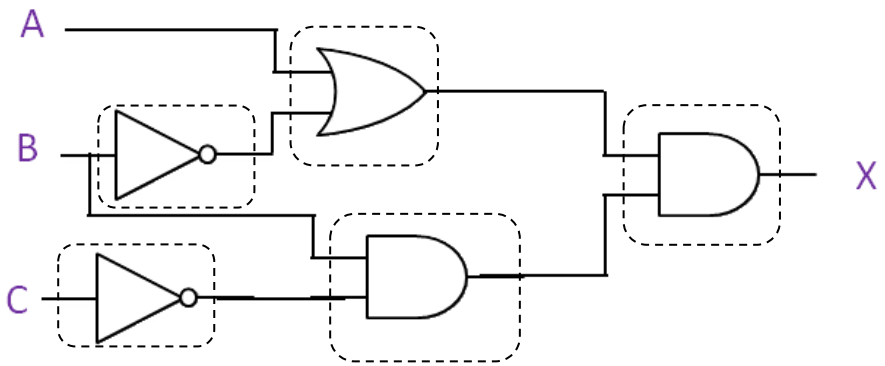
Question	Answer	Marks
4	**One** mark for each point **max three**. • variables / constants are used to store items of data • the data stored in variables / constants are accessed by an identifier // named data stores • the value of a variable may change during the execution of a program • the value of a constant will remain the same during the execution of a program	**3**

Question	Answer	Marks
5	**One** mark for a suitable hierarchical structure **One** mark for suitable names for the sub systems for user input and display options **One** mark for sub systems for user inputs, (choice of display,) food order and payment **One** mark for sub systems for display output types, pictures and list For example:  	**4**

Question	Answer	Marks

6(a)	**One** mark for each error identified and correction
	• Line `05 OUTPUT UsefulEnergyOut` should be `INPUT UsefulEnergyOut`
	• Line `06 IF TotalEnergyIn <> -1 AND UsefulEnergy <> -1` should be:
	`IF TotalEnergyIn <> -1 AND UsefulEnergyOut <> -1`
	• Line `11 UNTIL TotalEnergyIn <> -1 OR UsefulEnergyOut <> -1` should be:
	`UNTIL TotalEnergyIn = -1 OR UsefulEnergyOut = -1`
		3

6(b)	**One** mark for checking for >= 92
	One mark for outputting "**A-rated**" only if the condition is met
	For example
	`IF Efficiency >= 92`
	`  THEN`
	`  OUTPUT "A-rated"`
	`ENDIF`
		2

Question	Answer	Marks
7(a)	**One** mark for each correct gate, with the correct inputs as shown. 	**5**

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>7(b)</td><td>
<table>
<tr><th>A</th><th>B</th><th>C</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>0</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0</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>1</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>0</td></tr>
</table>
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
</td><td>4</td></tr>
</table>

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>8(a)</td><td>
<table>
<tr><th>NumberSales</th><th>Total</th><th>SaleValue</th><th>Average</th><th>OUTPUT</th></tr>
<tr><td>0</td><td>0</td><td></td><td></td><td></td></tr>
<tr><td>1</td><td>5.50</td><td>5.50</td><td></td><td></td></tr>
<tr><td>2</td><td>8.90</td><td>3.40</td><td></td><td></td></tr>
<tr><td>3</td><td>15.15</td><td>6.25</td><td></td><td></td></tr>
<tr><td>4</td><td>19.00</td><td>3.85</td><td></td><td></td></tr>
<tr><td>5</td><td>8.00</td><td>-11.00</td><td></td><td></td></tr>
<tr><td></td><td></td><td>0</td><td>1.6</td><td>Average sale value 1.6</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
</table>
One mark for each column NumberSales, Total, SaleValue
One mark for columns Average and OUTPUT
</td><td>4</td></tr>
<tr><td>8(b)</td><td>Error – one mark for identification of error
for example including negative numbers / not differentiation between negative and positive values
Correction
One mark for correct placement and one mark for appropriate action
For example – after the input box insert a decision box to reject negative numbers</td><td>3</td></tr>
</table>

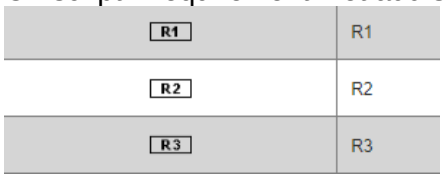
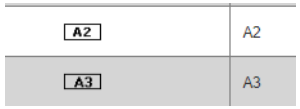
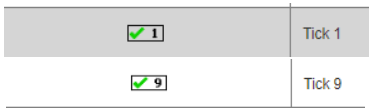
<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>9(a)</td><td>20</td><td>1</td></tr>
<tr><td>9(b)(i)</td><td>CatNo</td><td>1</td></tr>
<tr><td>9(b)(ii)</td><td>It is a unique identifier // no repeated values</td><td>1</td></tr>
<tr><td>9(c)</td><td>Two marks for 4 correct data types or one mark for 2 or 3 correct data types
<table>
<tr><th>Field</th><th>Data type</th></tr>
<tr><td>CatNo</td><td>Text/Alphanumeric</td></tr>
<tr><td>Title</td><td>Text/alphanumeric</td></tr>
<tr><td>Fiction</td><td>Boolean</td></tr>
<tr><td>Price</td><td>Real</td></tr>
</table></td><td>2</td></tr>
<tr><td>9(d)</td><td>One mark for each correct row
BK08 The Princesses’ Story B Penn
BK31 Networking for Beginners A Smith</td><td>2</td></tr>
<tr><td>9(e)</td><td>One mark if two correct or two marks if completely correct
Title
BookList
Author = "B Penn" // Author = 'B Penn' // Author Like "B Penn"</td><td>2</td></tr>
</table>

Question	Answer	Marks

10(a)	**One** mark for each correct line
	`DECLARE X : INTEGER`
	`DECLARE Y : REAL`
	`DECLARE Z : BOOLEAN`
		3

10(b)	**One** mark for using `FUNCTION` and `ENDFUNCTION` and `RETURNS BOOLEAN`
	One mark for naming the function `Same`
	One mark for defining the two parameters correctly
	One mark for comparing the two parameters using `ROUND`
	One mark for correctly returning `TRUE` and `FALSE`
	One mark for correct function call
	
	Example definition:
	`FUNCTION Same(A : INTEGER, B : REAL) RETURNS BOOLEAN`
	`  IF A = ROUND(B,0)`
	`  THEN`
	`  RETURN TRUE`
	`  ELSE`
	`  RETURN FALSE`
	`  ENDIF`
	`ENDFUNCTION`
	
	Example call:
	`Z ← Same(X,Y)`
		6

10(c)	A function is defined once **and** called many times **or** Define – setting up the function **and** call is using a function	**1**

Question	Answer	Marks
11	Read and understand the question before starting to mark any scripts. Read the whole answer before marking a script: 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 script if requirement met add seen, NE if partial attempt, cross if no attempt (see marked scripts).  Use the tables for A02 and A03 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 `TeamName`, `TeamPoints` Variables `LeagueSize`, `MatchNo` **Requirements (techniques)** **R1** calculates total points for all matches played by each team (nested iteration, totalling) **R2** counts and outputs, with the team's name, for each team the total number of away wins, home wins, drawn matches and lost matches (nested iteration, counting, output) **R3** finds and outputs the name of the team with the highest number of points and the name of the team with the lowest number of points. (output, selection)	**15**

Question	Answer	Marks

11	***Example 15-mark answer in pseudocode:***
	`// meaningful identifier names and appropriate data structures to store the data required`
	`DECLARE TeamCounter : INTEGER`
	`DECLARE MatchCounter : INTEGER`
	`FOR TeamCounter ← 1 to LeagueSize // zero totals for each club's results`
	`  TotalPoints[TeamCounter] ← 0`
	`NEXT TeamCounter`
	
	`FOR TeamCounter ← 1 TO LeagueSize`
	`  AwayWinNo ← 0  // zero totals for each club's result details`
	`  HomeWinNo ← 0`
	`  DrawNo ← 0`
	`  LostNo ← 0`
	`  FOR MatchCounter ← 1 TO MatchNo`
	`  TotalPoints[TeamCounter] ← TotalPoints[TeamCounter] +`
	`  TeamPoints[TeamCounter, MatchCounter]`
	`  CASE OF TeamPoints[TeamCounter, MatchCounter]`
	`  3 : AwayWinNo ← AwayWinNo + 1`
	`  2 : HomeWinNo ← HomeWinNo + 1`
	`  1 : DrawNo ← DrawNo + 1`
	`  0 : LostNo ← LostNo + 1`
	`  ENDCASE`
	`  NEXT MatchCounter`
	
	`  OUTPUT "Team ", TeamName[TeamCounter]  // Output details of a team's results`
	`  OUTPUT "Total points ", TotalResult[TeamCounter]`
	`  OUTPUT "Away wins ", AwayWinNo`
	`  OUTPUT "Home wins ", HomeWinNo`
	`  OUTPUT "Draws ", DrawNo`
	`  OUTPUT "Losses ", LostNo`	

Question	Answer	Marks

11	`// Check for highest and lowest results`
	`  IF TeamCounter = 1`
	`  THEN`
	`  HighestResult ← TotalPoints[TeamCounter]`
	`  LowestResult ← TotalPoints[TeamCounter]`
	`  ENDIF`
	
	`  IF TotalPoints[TeamCounter] > HighestResult`
	`  THEN`
	`  HighestResult ← TotalPoints[TeamCounter]`
	`  TopTeam ← TeamCounter`
	`  ENDIF`
	`  IF TotalPoints[TeamCounter] < LowestResult`
	`  THEN`
	`  LowestResult ← TotalPoints[TeamCounter]`
	`  BottomTeam ← TeamCounter`
	`  ENDIF`
	
	`NEXT TeamCounter`
	
	`// output names of the teams with the highest and lowest number of points`
	`OUTPUT "Top Team ", TeamName[TopTeam]`
	`OUTPUT "Bottom Team ", TeamName[BottomTeam]`	

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.*