Cambridge International Examinations

Cambridge International General Certificate of Secondary Education

COMPUTER SCIENCE **0478/22**

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

MARK SCHEME

Maximum Mark : 75

Cambridge Assessment International Education – 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).

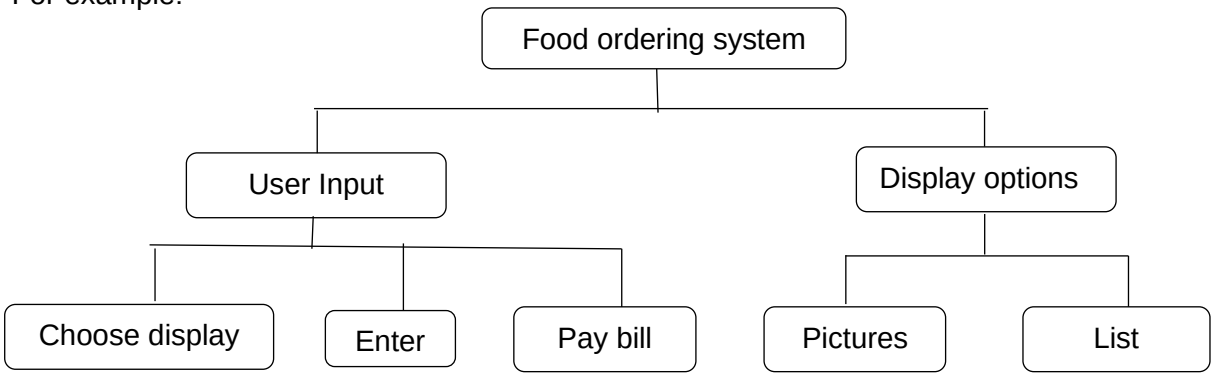
GENERIC MARKING PRINCIPLE 6:
Marks awarded are based solely on the requirements as defined in the mark scheme. Marks should not be awarded with grade thresholds or grade descriptors in mind.

Question	Answer	Marks	Guidance
1	**C**	1	
2(a)	**One** mark for each correct line **Description** / **Pseudocode statement** a statement to count → `Value ← Value + 1` a statement to total → `Value ← Value + NewValue` a statement to start a pre-condition loop → `WHILE Value > 10 DO` a statement to start a post-condition loop → `REPEAT` `FOR Count ← 1 TO 10`	4	

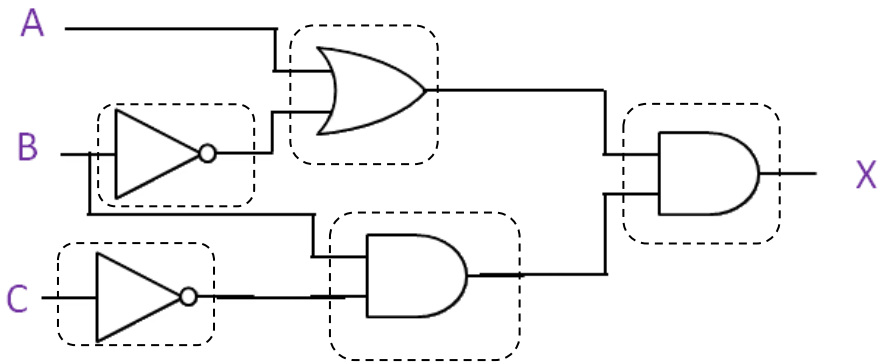
2(b)	**One** mark for each point:
	• Initialisation
	• appropriate loop controls
	• totalling statement inside the loop
	• calculation of average outside loop
	• output of average outside loop.
	Example:
	`Total ← 0`
	`FOR Count ← 1 TO 50`
	`  Total ← Total + Number[Count]`
		5	

Question	Answer	Marks	Guidance
	NEXT Count Average ← Total / 50 OUTPUT "The average is ", Average		

Question	Answer	Marks	Guidance
3	**One** mark for description, **one** mark for expansion and **one** mark for example e.g. To ensure that the program works as expected and rejects any invalid data that is input. Normal data (to check the results of a calculation)	**3**	•
4	**One** mark for each point **max three**. • variables and constants are used to store single items of data • the data stored in variables and constants are accessed by identifier • variables may change during the execution of a program // can answer by example • constants will remain the same during the execution of a program // can answer by example	**3**	for discussion at the STM • It could be argued that some variables store more than a single item of data. • The last word in the second bullet point should perhaps be '*identifier(s)*'.
5	**One** mark for a hierarchical structure **One** mark for suitable names for the sub systems **One** mark for identifiable inputs **One** mark for identifiable outputs	**4**	**STM** was meant by '*identifiable inputs*' and '*identifiable outputs*'. Some examples might be appreciated by examiners

Question	Answer	Marks	Guidance
	For example:  		

Question	Answer	Marks	Guidance
6(a)	**One** mark for each error identified and correction • Line `05 OUTPUT` should be `INPUT` • Line `06 UsefulEnergy` should be `UsefulEnergyOut` 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**	

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th><th>Guidance</th></tr>
<tr><td>7(a)</td><td>One mark for each correct gate, with the correct input(s) as shown.
 
</td><td>5</td><td></td></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><td></td></tr>
</table>

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th><th>Guidance</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 <code>Total</code>, <code>Average</code> and <code>OUTPUT</code>
One mark for columns <code>NumberSales</code> and <code>SaleValue</code>
</td><td>4</td><td></td></tr>
<tr><td>8(b)</td><td>Error – including negative numbers / not differentiation between negative and positive values

Correction
One mark for placement and one mark action</td><td>3</td><td>STM discuss refunds</td></tr>
</table>

Question	Answer	Marks	Guidance
	For example – after the input box insert a decision box to reject negative numbers		

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th><th>Guidance</th></tr>
<tr><td>9(a)</td><td>20</td><td>1</td><td></td></tr>
<tr><td>9(b)(i)</td><td>CatNo</td><td>1</td><td></td></tr>
<tr><td>9(b)(ii)</td><td>It is a unique identifier</td><td>1</td><td></td></tr>
<tr><td>9(c)</td><td>2 marks for 4 correct data types

1 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><td></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><td>Any errors in order penalise once and follow through</td></tr>
<tr><td>9(e)</td><td>One mark if two correct or two marks if completely correct
Title
BookList
Author = "B Penn"</td><td>2</td><td></td></tr>
</table>

Question	Answer	Marks	Guidance

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 and using `ROUND`
	One mark for correctly returning `TRUE` and `FALSE`
	One mark for correct function call
	
	For example definition:
	`FUNCTION Same(A : INTEGER, B : REAL) RETURNS BOOLEAN`
	`  IF A = ROUND(B,0)`
	`  THEN`
	`  RETURN TRUE`
	`  ELSE`
	`  RETURN FALSE`
	`  ENDIF`
	`ENDFUNCTION`
	
	For 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 the whole answer: Check if each requirement listed below has been met. Requirements may be met using a suitable built-in function from the programming language used (probably Python, VB.NET or Java) On script tick if requirement met, cross if no attempt seen, omission mark and/or comment if partially met (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** names shown underlined must be used as given in the scenario Arrays or lists `TeamName`, `TeamPoints`, (`TotalPoints`, `TotalAwayWin`, `TotalHomeWin`, `TotalDraw` and `TotalLost` may be seen but no requirement to store) Variables `LeagueSize`, `MatchNo` Constants `AwayWin`, `HomeWin`, `Draw`, and `Loss` could be variables **Requirements (techniques)** **R1** calculates total result for all matches played by each team (nested iteration and totalling) **R2** counts the total number of away wins, home wins, draw matches and lost matches for each team (nested iteration and counting) **R3** outputs for each team name, total result, total number of away wins, home wins, drawn matches and lost matches (output and selection) **R4** finds and outputs the name of the team with the highest result and the team with the lowest result. (output and selection)	**15**

Question	Answer	Marks
	Example 15 mark answer in pseudocode:	For information inline with instruction on the QP

```
// meaningful identifier names and appropriate data structures to store the data required
DECLARE TeamCounter : INTEGER
DECLARE MatchCounter : INTEGER
DECLARE TeamPoints : INTEGER
DECLARE AwayWinNo : INTEGER
DECLARE HomeWinNo : INTEGER
DECLARE DrawNo : INTEGER
DECLARE LostNo : INTEGER
DECLARE HighestPoints : INTEGER
DECLARE LowestPoints : INTEGER
DECLARE TopTeam : INTEGER
DECLARE BottomTeam : INTEGER


CONSTANT AwayWin = 3
CONSTANT HomeWin = 2
CONSTANT Draw = 1
CONSTANT Lost = 0

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]
```

Question	Answer	Marks

```
            AwayWin : AwayWinNo ← AwayWinNo + 1
            HomeWin : HomeWinNo ← HomeWinNo + 1
            Draw : DrawNo ← DrawNo + 1
            Lost : 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 ", LossNo

// 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
```

Question	Answer	Marks
	NEXT TeamCounter OUTPUT "Top Team ", TeamName[TopTeam] // output highest and lowest team results 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.*