Cambridge Assessment International Education

Cambridge IGCSE™

COMPUTER SCIENCE **0478/23**

Paper 2 Algorithms, Programming and Logic **October/November 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 October/November 2023 series for most Cambridge IGCSE, Cambridge International A and AS Level components, and some Cambridge O Level components.

This document consists of **16** printed pages.

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.

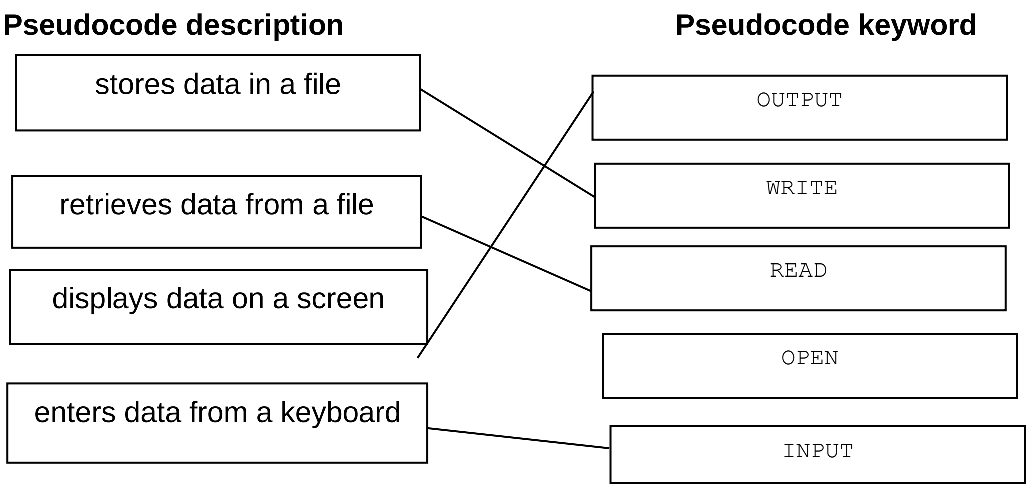
GENERIC MARKING PRINCIPLE 4:

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
1	**A**	**1**

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

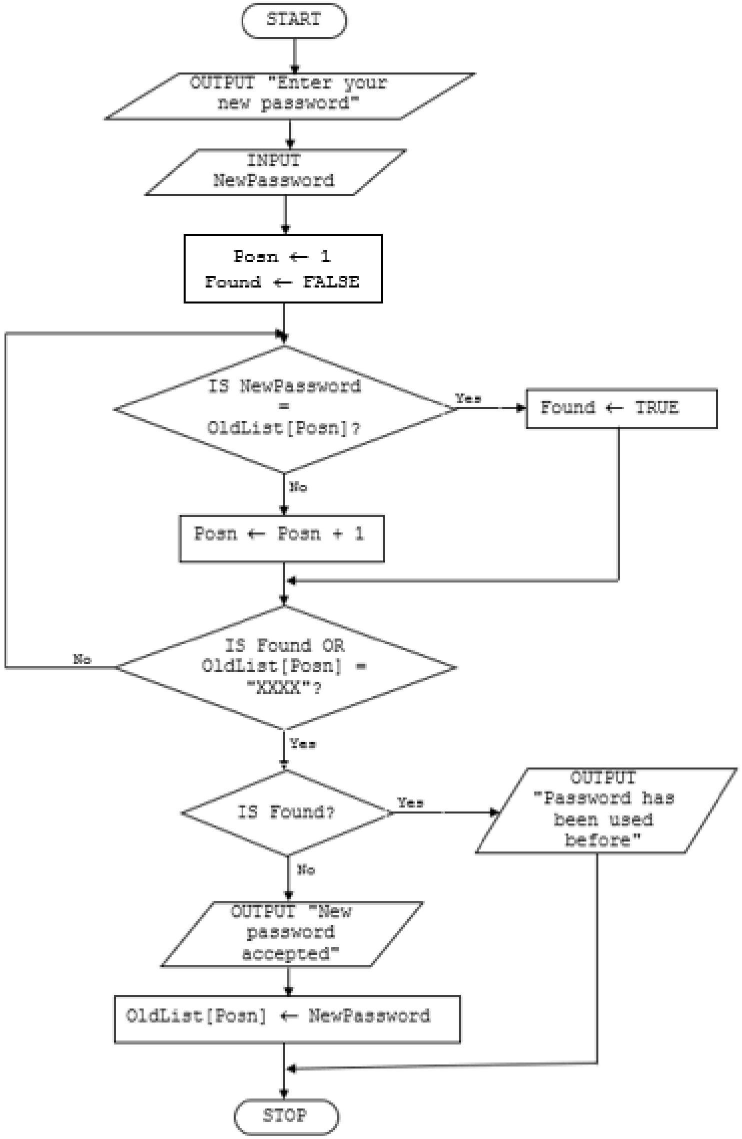
Question	Answer	Marks
3(a)	**One** mark for each correct line from description to pseudocode keyword  Pseudocode description stores data in a file retrieves data from a file displays data on a screen enters data from a keyboard Pseudocode keyword OUTPUT WRITE READ OPEN INPUT	**4**
3(b)	**One** mark for each point (max two) • data is stored permanently • data can be moved to another computer • another copy of data can be made and stored//accessed elsewhere // backup copy	**2**

Question	Answer	Marks
4(a)	**One** mark for each point • type check • range check	**2**

4(b)	**One** mark for each point (max five)
	• use of loop for check
	• checking for whole number
	• checking for number greater than or equal to one
	• … and less than or equal to six
	• Appropriate error/reinput message
	• ability to reinput value
	
	Example:
	`WHILE Seats < 1 OR Seats > 6 OR Seats <> ROUND(Seats, 0) DO`
	`    OUTPUT "Please enter a valid number of seats "`
	`    INPUT Seats`
	`ENDWHILE`
		5

4(c)	**One** mark for correct test data, **one** mark for corresponding reason Example: 7, abnormal data to show that this value would be rejected	**2**

Question	Answer	Marks
5(a)	**One** mark for each error identified and correction given • Line `06 Password` should be `NewPassword` • Line `11 AND` should be `OR` • Line `16 INPUT` should be `OUTPUT`	**3**

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>5(b)</td><td>

START
OUTPUT "Enter your new password"
INPUT NewPassword
Posn ← 1
Found ← FALSE
IS NewPassword = OldList[Posn]?
Yes
Found ← TRUE
No
Posn ← Posn + 1
IS Found OR OldList[Posn] = "XXXX"?
No
Yes
IS Found?
Yes
OUTPUT "Password has been used before"
No
OUTPUT "New password accepted"
OldList[Posn] ← NewPassword
STOP

Max six marks from:

Max four from:
one mark for data entry with message
one mark for initialisation
one mark for checking list // decision box comparing input with array
one mark for updating // updating the two variables position and found
one mark for loop control // second decision box
one mark for setting new password to position in list
one mark for outputs // two outputs

Two marks:
one mark for correct use of flow chart symbols
one mark for correct use arrows and labels
</td><td>6</td></tr>
</table>

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>6(a)</td><td>

One mark for correct gate and **one** mark for correct truth table

AND

<table>
<tr><th>A</th><th>B</th><th>X</th></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>1</td></tr>
</table>

</td><td>2</td></tr>
<tr><td>6(b)</td><td>

One mark for correct gate and **one** mark for correct truth table

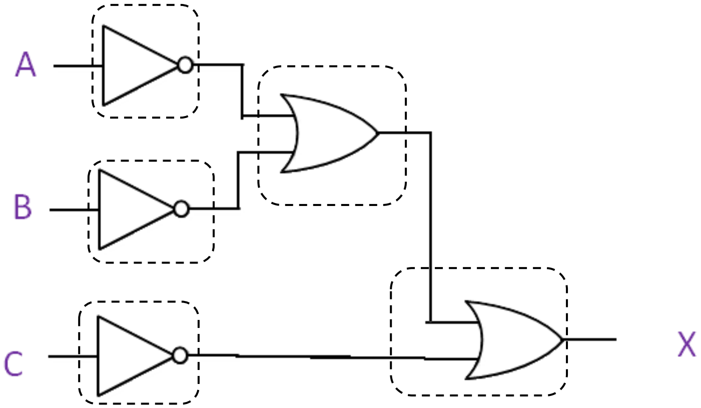
XOR // EOR

<table>
<tr><th>A</th><th>B</th><th>X</th></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>0</td></tr>
</table>

</td><td>2</td></tr>
</table>

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>6(c)</td><td><b>One</b> mark for correct gate and <b>one</b> mark for correct truth table

<b>NOR</b>

<table>
<tr><th>A</th><th>B</th><th>X</th></tr>
<tr><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>0</td></tr>
</table>
</td><td><b>2</b></td></tr>
<tr><td>6(d)</td><td><b>One</b> mark for each correct gate, with the correct input(s) as shown.


</td><td><b>5</b></td></tr>
</table>

Question	Answer	Marks
7	**one** mark for first description **one** mark for matching difference **max four** • local variables - scope is a defined block of code/subroutine/procedure/function • global variables – scope is the whole program • local variables - value cannot be changed elsewhere in the program • global variables – value can be changed anywhere in the program	**4**

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>8(a)</td><td>
<table>
<tr><th>Accept</th><th>Reject</th><th>PartOK</th><th>Error</th><th>OUTPUT</th></tr>
<tr><td>0</td><td>0</td><td></td><td></td><td></td></tr>
<tr><td>1</td><td></td><td>Y</td><td></td><td></td></tr>
<tr><td>2</td><td></td><td>Y</td><td></td><td></td></tr>
<tr><td>3</td><td></td><td>Y</td><td></td><td></td></tr>
<tr><td></td><td>1</td><td>N</td><td></td><td></td></tr>
<tr><td>4</td><td></td><td>Y</td><td></td><td></td></tr>
<tr><td>5</td><td></td><td>Y</td><td></td><td></td></tr>
<tr><td>6</td><td></td><td>Y</td><td></td><td></td></tr>
<tr><td>7</td><td></td><td>Y</td><td></td><td></td></tr>
<tr><td></td><td>2</td><td>N</td><td></td><td></td></tr>
<tr><td>8</td><td></td><td>Y</td><td></td><td></td></tr>
<tr><td>9</td><td></td><td>Y</td><td></td><td></td></tr>
<tr><td>10</td><td></td><td>Y</td><td>20</td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td>Too many rejected 20% error</td></tr>
</table>
<b>One</b> mark for each column
</td><td>5</td></tr>
</table>

Question	Answer	Marks
8(b)	**One** mark for each point **max three** • after the Input box // before the first decision box • insert a process box • to convert the input to upper case **OR** • change the first decision / add another decision box • to accept `'y'` as well • by adding `OR PartOK = 'y'`	**3**

Question	Answer	Marks
9(a)	Records: 14 Fields: 5	**2**
9(b)(i)	Species/Description	**1**
9(b)(ii)	Long names that could be easily misspelt // species or description could be duplicated	**1**
9(b)(iii)	Easy to validate // always unique	**1**
9(c)	**One** mark for each correct row **or** column True silver white laced top half and black lower half Brown eared brown with ear tufts	**2**

9(d)	**One** mark for each correct addition
	`SELECT Species`
	`FROM PheasantList`
	`WHERE Breeding` **or** `WHERE Young = 0`
	`AND Young = 0;` `AND Breeding;`
		4

Question	Answer	Marks

10	• AO2 (maximum 9 marks)
	• AO3 (maximum 6 marks)
	
	Data Structures required names shown underlined must be used as given in the scenario
	2D Array or list `Temperatures`
	Variables `MaxDay`, `MinDay`, `AvDay`, `MaxWeek`, `MinWeek`, `AvWeek`
	
	Requirements (techniques)
	
	R1 Find maximum and minimum temperatures for each day and calculates the average daily temperature (searching, totalling)
	R2 Find maximum and minimum temperatures for week and calculates the average weekly temperature (nested searching, totalling)
	R3 outputs for each day name, the rounded values for maximum temperature, minimum temperatures and average temperature. Outputs for the week the rounded values for maximum temperature, minimum temperatures and average temperature (output with appropriate messages and rounded values)
	
	Example 15-mark answer in pseudocode:
	
	`// meaningful identifier names and appropriate data structures to store the data required`
	`DECLARE DayCounter, HourCounter : INTEGER`
	`DECLARE AvDay, AvWeek, MaxDay, MinDay, MaxWeek, MinWeek : REAL`
	`DECLARE DayTotal, WeekTotal : REAL`
	`DECLARE Day : STRING`
	
	`CONSTANT Hours ← 24`
	`CONSTANT Days ← 7`
		15

Question	Answer	Marks
10		

```
MaxWeek ← -1000// initialise max and min temperatures and total for the week
MinWeek ← 1000
WeekTotal ← 0

FOR DayCounter ← 0 TO Days - 1
    MaxDay ← -1000// initialise max and min temperatures and total for each day
    MinDay ← 1000
    DayTotal ← 0
    FOR HourCounter ← 0 TO Hours - 1
        DayTotal ← DayTotal + Temperatures(HourCounter, DayCounter)
          // update total maximum and minimum
        IF Temperatures(HourCounter, DayCounter) > MaxDay
          THEN
            MaxDay ← Temperatures(HourCounter, DayCounter)
        ENDIF
        IF Temperatures(HourCounter, DayCounter) < MinDay
          THEN
            MinDay ← Temperatures(HourCounter, DayCounter)
        ENDIF
    NEXT HourCounter

    CASE OF DayCounter  // select message for day
      0 : Day ← "Monday"
      1 : Day ← "Tuesday"
      2 : Day ← "Wednesday"
      3 : Day ← "Thursday"
      4 : Day ← "Friday"
      5 : Day ← "Saturday"
      6 : Day ← "Sunday"

    ENDCASE

    DayAverage ← DayTotal / Hours  // output results for day
    OUTPUT Day  // Results from a day
    OUTPUT "Maximum temperature ", MaxDay
    OUTPUT "Minimum temperature ", MinDay
    OUTPUT "Average temperature ", ROUND(DayAverage,2)
```

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>10</td><td>

```
    IF MaxDay > MaxWeek  // update total maximum and minimum
      THEN
        MaxWeek ← MaxDay
    ENDIF

    IF MinDay > MinWeek
      THEN
        MinWeek ← MinDay
    ENDIF
    WeekTotal ← WeekTotal + DayTotal  // update total for week

NEXT DayCounter
    WeekAverage ← WeekTotal / Days

    OUTPUT "Maximum temperature for week ", MaxWeek// output results for week
    OUTPUT "Minimum temperature for week ", MinWeek
    OUTPUT "Average temperature for Week ", ROUND(WeekAverage,2)
```

</td><td></td></tr>
</table>

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 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 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.*