Cambridge Assessment International Education

Cambridge O Level

COMPUTER SCIENCE **2210/22**

Paper 2 Algorithms, Programming and Logic **May/June 2023**

MARK SCHEME

Maximum Mark: 50

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 May/June 2023 series for most Cambridge IGCSE, Cambridge International A and AS Level and Cambridge Pre-U components, and some Cambridge O Level components.

© UCLES 2023

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.

© UCLES 2023

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.

Please note the following further points:

The words in **bold** in the mark scheme are important text that needs to be present, or some notion of it needs to be present. It does not have to be the exact word, but something close to the meaning.

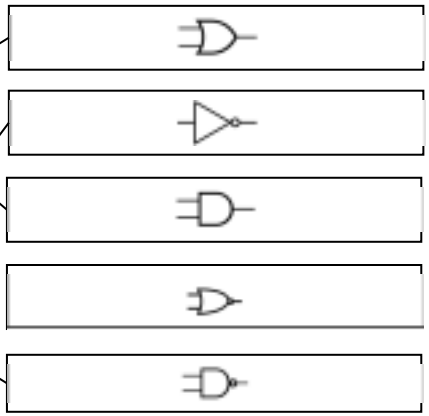
If a word is underlined, this **exact** word must be present.

A single forward slash means this is an alternative word. A double forward slash means that this is an alternative mark point.

Ellipsis (…) on the end of one-mark point and the start of the next means that the candidate **cannot** get the second mark point without being awarded the first one. If a mark point has an ellipsis at the beginning, but there is no ellipsis on the mark point before it, then this is just a follow-on sentence and **can** be awarded **without** the previous mark point.

© UCLES 2023

Question	Answer	Marks
1	**A**	**1**

Question	Answer	Marks
2	**One** mark for each correct line **Logic gate** / **Standard symbol** AND OR NAND NOT 	**4**

Question	Answer	Marks
3	**One** mark for each correct answer • structure diagram / chart • flowchart • pseudocode	**3**

© UCLES 2023

Question	Answer	Marks
4(a)	**One** mark for each point (**max three**). • range check with acceptable values is (greater than) zero **and** less than 1000 • presence check to ensure the program will not continue until a value has been entered • type/character check to ensure that a number is entered • length check to ensure there are no more than 3 digits entered	**3**
4(b)(i)	To verify the data / for verification / as a verification check // to make sure that no changes are made to the data on entry	**1**

4(b)(ii)	**One** mark for each point (**max three**).
	• use of iteration
	• use of two inputs
	• to check that the two inputs are the same / different
	• use of the given variable `Measurement`
	
	For example
	`REPEAT`
	`  OUTPUT "Please enter measurement "`
	`  INPUT Measurement`
	`  OUTPUT "Please re-enter measurement "`
	`  INPUT MeasurementCheck`
	`UNTIL Measurement = MeasurementCheck`
		3

Question	Answer	Marks
5	Due to an issue with Question 5, the question has been removed from the question paper.	

© UCLES 2023

Question	Answer	Marks
6	**One mark** for each feature and **one mark** for corresponding example (**max six**) • ensuring that all identifiers have meaningful names … • … example using `Total` to store a running total • using comments to explain how the program works … • … example `// all values are zeroed before the next calculation` • using procedures and functions for the tasks within a program … • … example `CalculateInterest(Deposit, Rate)`	**6**

Question	Answer	Marks
7(a)	• `07` • `04/12` or `16/18` • `02/20`	**3**
7(b)	**One** mark for each error identified and correction • Line `07 Total ← Total + Number * Counter` should be `Total ← Total + Number[Counter] * Counter` • Line `08 IF Number[Counter] = 0` should be `IF Number[Counter] = -1` // should be `IF Number[Counter] < 0` • Line `16 FOR Counter ← 0 TO 5` should be `FOR Counter ← 1 TO 5`	**3**

Question	Answer	Marks

7(c)	**One** mark for place in algorithm (**max one**)
	• around lines `05` and `06`
	• line `07`
	• (immediately) after the input of the number
	
	Three marks pseudocode
	One mark for each point (**max three**)
	• Use of `REPEAT … UNTIL` // any working loop structure
	• check for `>0` // `>=0`
	• check for `<10` // `>9`
	• check for whole number
	• check for -1
	• check for length of digit <> 1
	
	Example
	`REPEAT`
	`  OUTPUT "Enter a digit "`
	`  INPUT Number[Counter]`
	`UNTIL Number[Counter] = Round(Number[Counter],0) AND ((Number[Counter] = -1) OR`
	`  (Number[Counter] > 0 AND Number[Counter] < 10))`
		4

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>8</td><td>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

<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>1</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>0</td></tr>
</table>
</td><td>4</td></tr>
</table>

© UCLES 2023

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>9(a)</td><td>
One mark for each column <code>F</code>, <code>C</code> and <code>T</code>
Two marks for columns <code>X[1]</code> to <code>X[5]</code> all entries correct or
One mark for columns <code>X[1]</code> to <code>X[5]</code> with one error

<table>
<tr><th>F</th><th>C</th><th>X[1]</th><th>X[2]</th><th>X[3]</th><th>X[4]</th><th>X[5]</th><th>T</th></tr>
<tr><td></td><td></td><td>10</td><td>1</td><td>5</td><td>7</td><td>11</td><td></td></tr>
<tr><td>0</td><td>1</td><td></td><td></td><td></td><td></td><td></td><td>10</td></tr>
<tr><td>1</td><td>2</td><td>1</td><td>10</td><td></td><td></td><td></td><td>10</td></tr>
<tr><td>1</td><td>3</td><td></td><td>5</td><td>10</td><td></td><td></td><td>10</td></tr>
<tr><td>1</td><td>4</td><td></td><td></td><td>7</td><td>10</td><td></td><td></td></tr>
<tr><td></td><td>5</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>0</td><td>1</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>2</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>3</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>4</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>5</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>
</td><td>5</td></tr>
<tr><td>9(b)</td><td>
One mark for each point
• (bubble) sort data in array
• in ascending order
</td><td>2</td></tr>
</table>

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>10(a)</td><td>SongNumber</td><td>1</td></tr>
<tr><td>10(b)</td><td>One mark for every two correct data types

<table><tr><th>Field</th><th>Data Type</th></tr><tr><td>SongNumber</td><td>Text/Alphanumeric</td></tr><tr><td>Title</td><td>Text/Alphanumeric</td></tr><tr><td>Recorded</td><td>Date/time</td></tr><tr><td>Minutes</td><td>Real</td></tr></table></td><td>2</td></tr>
<tr><td>10(c)</td><td>One mark for each point
• to find the total number of minutes of music
• to find the total number of songs
• available for the genre rock</td><td>3</td></tr>
</table>

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>11(a)</td><td>One mark for any two correct lines

DECLARE P : STRING
P ← "The world"
DECLARE Q : CHAR
Q ← 'W'</td><td>2</td></tr>
</table>

Question	Answer	Marks

11(b)	**One** mark for each point (**max four**)
	• converting `P` to upper case
	• finding the length of `P`
	• using a loop to check for position of `Q`
	• using the string operation substring
	• storing the loop counter in `Position` if the value is found
	
	For example:
	`P ← UCASE(P)`
	`Counter ← 1`
	`Position ← 0`
	`REPEAT`
	`  IF SUBSTRING(P, Counter, 1) = Q`
	`  THEN`
	`  Position ← Counter`
	`  ENDIF`
	`  Counter ← Counter + 1`
	`UNTIL Position <> 0 OR Counter = LENGTH(P)`
		4

11(c)	5	**1**

Question	Answer	Marks

12	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 (Python, VB.NET or Java)
	Mark SEEN on script if requirement met, cross if no attempt seen, NE 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 `Account, AccDetails`
	Variable `Size, AccountNumber`
	
	Requirements (techniques)
	R1 Check account number and password (iteration and validation, selection, input, output)
	R2 Display menu and make a selection (output, input and selection)
	R3 Perform actions selected (use of arrays and procedures with parameters)
	
	Example 15 mark answer in pseudocode
	`// Procedures to be called`
	`PROCEDURE CheckDetails(AccID : INTEGER)`
	`  DECLARE Name, Password : STRING // local variables`
	`  Valid ← FALSE`
	`  IF AccID <0 OR AccID > Size`
	`  THEN`
	`  OUTPUT "Invalid Account Number"`
	`  ELSE`
	`  OUTPUT "Please Enter Name "`
	`  INPUT Name`
	`  OUTPUT "Please Enter Password "`
	`  INPUT Password`
	`  IF Name <> Account[AccID,1] OR Password <> Account[AccID,2]`
	`  THEN`
	`  OUTPUT "Invalid name or password"`
	`  ELSE`
		15

Question	Answer	Marks

12	Valid ← True
	ENDIF
	ENDIF
	ENDPROCEDURE
	
	PROCEDURE Balance(AccID : INTEGER)
	OUTPUT "Your balance is ", AccDetails[AccID,1]
	ENDPROCEDURE
	
	PROCEDURE WithDrawal(AccID : INTEGER)
	DECLARE Amount : REAL // local variable
	REPEAT
	OUTPUT "Please enter amount to withdraw "
	INPUT Amount
	IF Amount > AccDetails[AccID,3]
	THEN
	OUTPUT "Amount greater than withdrawal limit"
	ENDIF
	IF Amount > AccDetails[AccID,2] + AccDetails[AccID,1]
	THEN
	OUTPUT "Amount greater than cash available"
	ENDIF
	IF Amount <= AccDetails[AccID,3] AND Amount < AccDetails[AccID,2] +
	AccDetails[AccID,1]
	THEN
	AccDetails[AccID,1] ← AccDetails[AccID,1] - Amount
	ENDIF
	UNTIL Amount<= AccDetails[AccID,3] AND Amount > AccDetails[AccID,2] +
	AccDetails[AccID,1] AND Amount > 0
	ENDPROCEDURE
	
	PROCEDURE Deposit(AccID : INTEGER)
	DECLARE Amount : REAL // local variable
	REPEAT
	OUTPUT "Please enter a positive amount to deposit "
	INPUT Amount
	UNTIL Amount >0
	AccDetails[AccID,1] ← AccDetails[AccID,1] + Amount	

© UCLES 2023

Question	Answer	Marks

12	ENDPROCEDURE
	
	// Declarations of global variables for information – not required in candidate responses
	DECLARE AccountNumber, Choice : INTEGER
	DECLARE Valid, Exit : BOOLEAN
	
	OUTPUT "Please enter your account number "
	INPUT AccountNumber
	CheckDetails(AccountNumber)
	
	IF Valid
	  THEN
	  REPEAT
	  OUTPUT "Menu"
	  OUTPUT "1. display balance"
	  OUTPUT "2. withdraw money"
	  OUTPUT "3. deposit money"
	  OUTPUT "4. exit"
	  OUTPUT "please choose 1, 2, 3 or 4"
	  INPUT Choice
	  CASE OF Choice
	  1 : Balance(AccountNumber)
	  2 : Withdrawal(AccountNumber)
	  3 : Deposit(AccountNumber)
	  4 : Exit ← TRUE
	  OTHERWISE OUTPUT "Invalid choice"
	  ENDCASE
	  UNTIL Choice = 4
	  ELSE
	  OUTPUT "Invalid account number "
	ENDIF	

<table>
<tr><th colspan="4">Marking Instructions in italics</th></tr>
<tr><th colspan="4">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</th></tr>
<tr><th>0</th><th>1–3</th><th>4–6</th><th>7–9</th></tr>
<tr><td rowspan="2">No creditable response.</td><td>At least one programming technique has been used.

Any use of selection, iteration, counting, totalling, input and output.</td><td>Some programming techniques used are appropriate to the problem.

More than one technique seen applied to the scenario, check the list of techniques needed.</td><td>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.</td></tr>
<tr><td>Some data has been stored but not appropriately.

Any use of variables or arrays or other language dependent data structures e.g. Python lists.</td><td>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.</td><td>The data structures chosen are appropriate and store all the data required.

The data structures used store all the data required by the scenario.</td></tr>
</table>

© UCLES 2023

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.*

© UCLES 2023