Cambridge Assessment International Education

Cambridge IGCSE™ (9–1)

COMPUTER SCIENCE **0984/21**

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.

[Turn over

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:

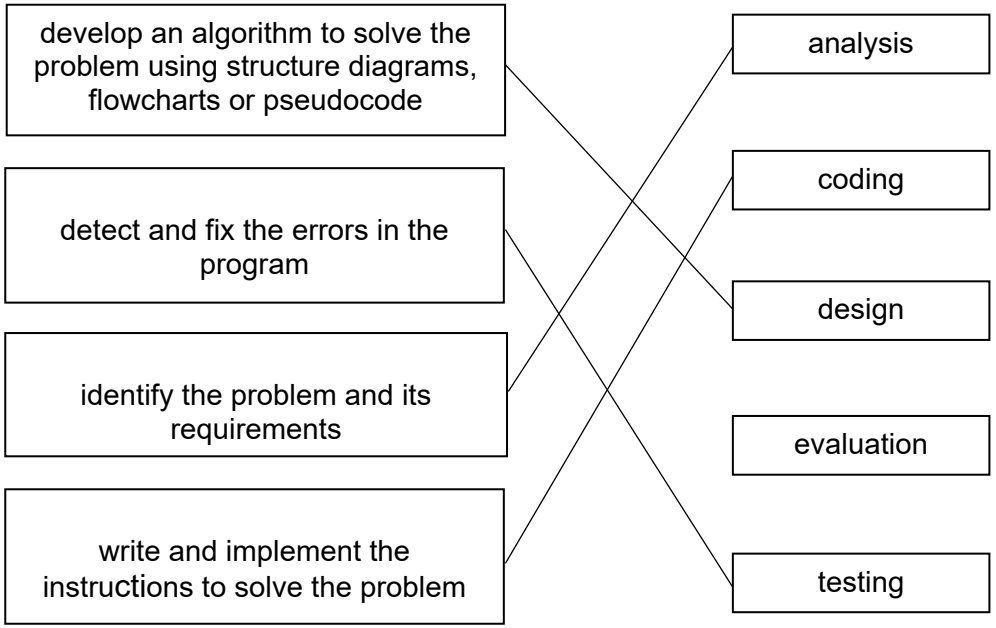
Marks must be awarded **positively**:

- 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.

Question	Answer	Marks
1(a)	**One** mark for each correct line. **Program development life cycle description** **Program development life cycle stage**  	**4**
1(b)	**One** mark for naming or describing each component part, max **three** **For example:** inputs // what is put into the system processes // actions taken to achieve a result outputs // what is taken out of the system storage // what needs to be kept for future use	**3**

Question	Answer	Marks
2	A	**1**

Question	Answer	Marks
3(a)	**One** mark per mark point, max **two** • Validation is an automated check carried out by a computer • … to make sure the data entered is sensible/acceptable/reasonable	**2**
3(b)	**One** mark for each appropriate test data, max **three** **One** mark for each correct accompanying reason, max **three** **For example:** Normal – 75 Reason – the data lies within the required range **and** should be accepted Abnormal – Sixty Reason – this is the wrong data type **and** should be rejected Extreme – 200 Reason – the highest value in the required range that should be accepted	**6**

Question	Answer	Marks
4	**One** mark per mark point, max **four** `DIV`, max **two** • To perform integer division • Meaning only the whole number part of the answer is retained • Example of `DIV` For example `DIV(9,4) = 2` `ROUND`, max **two** • To return a value rounded to a specified number of digits / decimal places • The result will either be rounded to the next highest or the next lowest value • … depending on whether the value of the preceding digit is >=5 or <5 • Example of `ROUND` for example, `ROUND(4.56, 1) = 4.6`	**4**

Question	Answer	Marks

5(a)	**One** mark per mark point, max **four**
	
	• Line 04 / `IF Number < 0`
	should be `IF Number > 0`
	
	• Line 10 / `Exit ← 1` // Line 01/ `Exit ← 1` **and** Line 02 / `WHILE Exit <> 0`
	should be `Exit ← 0` // should be `Exit ← 0` **and** `WHILE Exit = 0`
	
	• Line 13 / `ENDIF`
	should be `ENDWHILE`
	
	• Line 14 / `OUTPUT "The total value of your numbers is ", Number`
	should be `OUTPUT "The total value of your numbers is ", Total`
	
	Correct algorithm:
	
	`01 Exit ← 1`
	`02 WHILE Exit <> 0 DO`
	`03  INPUT Number`
	`04  IF Number > 0`
	`05  THEN`
	`06  Total ← Total + Number`
	`07  ELSE`
	`08  IF Number = 0`
	`09  THEN`
	`10  Exit ← 0`
	`11  ENDIF`
	`12  ENDIF`
	`13 ENDWHILE`
	`14 OUTPUT "The total value of your numbers is ", Total`
		4

Question	Answer	Marks
5(b)	**One** mark per mark point, max **four** • Initialise a new (counting) variable • … `Count ← 0` // to count the acceptable numbers • Insert a counting statement between lines 05 and 07 • … `Count ← Count + 1` • Add a new output after the loop/after line 13 / at the end (of the program) • … `OUTPUT Count`	**4**

Question	Answer	Marks
6	**One** mark for each correct feature, max **two** **One** mark for each correct accompanying reason, max **two** **For example:** Meaningful identifiers – to enable the programmer (or future programmers) to easily recognize the purpose of a variable / array / constant // to enable easy tracking of a variable / constant / array through the program Use of comments – to annotate each section of a program so that a programmer can find specific sections / so that the programmer knows the purpose of that section of code Procedures and functions – to make programs modular and easier to update / add functionality	**4**

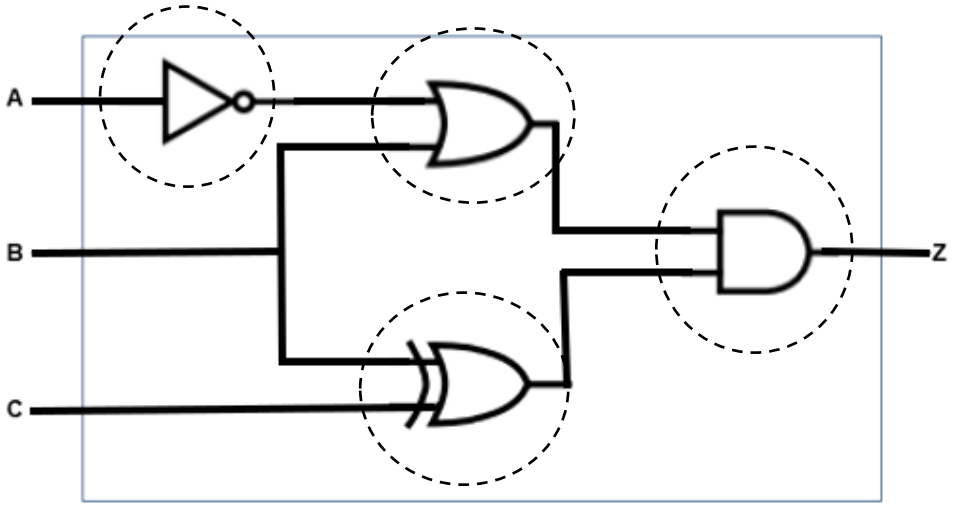
<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>7(a)</td><td>One mark per correct column, max four
<table>
<tr><th>Pointer</th><th>Letter</th><th>Choice</th><th>OUTPUT</th></tr>
<tr><td>1</td><td>F</td><td></td><td></td></tr>
<tr><td>2</td><td></td><td></td><td></td></tr>
<tr><td>3</td><td></td><td></td><td></td></tr>
<tr><td>4</td><td></td><td></td><td></td></tr>
<tr><td>5</td><td></td><td></td><td></td></tr>
<tr><td>6</td><td></td><td></td><td>Letter F is represented by Foxtrot</td></tr>
<tr><td></td><td></td><td></td><td>Another Letter? (Y or N)</td></tr>
<tr><td></td><td></td><td>Y</td><td></td></tr>
<tr><td>1</td><td>D</td><td></td><td></td></tr>
<tr><td>2</td><td></td><td></td><td></td></tr>
<tr><td>3</td><td></td><td></td><td></td></tr>
<tr><td>4</td><td></td><td></td><td>Letter D is represented by Delta</td></tr>
<tr><td></td><td></td><td></td><td>Another Letter? (Y or N)</td></tr>
<tr><td></td><td></td><td>N</td><td></td></tr>
</table></td><td>4</td></tr>
<tr><td>7(b)</td><td>(Linear) search</td><td>1</td></tr>
</table>

Question	Answer	Marks
7(c)	**One** mark per mark point, max **two** • The algorithm would not stop • … because it would not have found the item it was seeking **Or** • The array would run out of values after the pointer reached 13 • the algorithm will crash	**2**

Question	Answer	Marks

8(a)	**One** mark per mark point, max **three**
	• Storing string in `Phrase`
	• Correct use of `LENGTH` function
	• Correct use of `UCASE` function
	• Correct outputs of `LENGTH` and `UCASE`
	
	For example:
	
	`Phrase ← "The beginning is the most important part"`
	`OUTPUT LENGTH(Phrase)`
	`OUTPUT UCASE(Phrase)`
		3

8(b)	**One** mark for each correct line, max **two**
	
	`40`
	`THE BEGINNING IS THE MOST IMPORTANT PART`
		2

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

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>9(b)</td><td>Four marks for eight correct outputs.
Three marks for six or seven correct outputs.
Two marks for four or five correct outputs.
One mark for two or three correct outputs

<table>
<tr><th>A</th><th>B</th><th>C</th><th>Z</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>1</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1</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></td><td>4</td></tr>
</table>

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>10(a)</td><td>One mark for the correct field name
One mark for the correct reason

For example:

<code>TVCode</code>
Each entry in this field is a unique identifier</td><td>2</td></tr>
</table>

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>10(b)</td><td>Two marks for four correct answers.
One mark for two or three correct answers.

<table>
<tr><th>Field</th><th>Data type</th></tr>
<tr><td>TVCode</td><td>Text</td></tr>
<tr><td>ScreenSize</td><td>Integer</td></tr>
<tr><td>SmartTV</td><td>Boolean</td></tr>
<tr><td>Price$</td><td>Real</td></tr>
</table></td><td>2</td></tr>
<tr><td>10(c)</td><td>One mark for each correct answer

ScreenSize
Price$
FROM
YES

Correct code:

SELECT TVCode, ScreenSize, Price$
FROM TVRange
WHERE SmartTV = YES;</td><td>4</td></tr>
</table>

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 (Python, VB.NET or Java). On place a SEEN mark if requirement met, cross if no attempt seen, omission mark and/or comment if partially met (see marked scripts). 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: • AO2 (maximum 9 marks) • AO3 (maximum 6 marks) **Data structures required:** The names underlined must match those given in the scenario: Arrays or lists `Days[]`, `Readings[]`, `AverageTemp[]` Variables `WeekLoop`, `DayLoop`, `InTemp`, `TotalDayTemp`, `TotalWeekTemp`, `AverageWeekTemp` **Requirements (techniques):** **R1** Input and store hourly temperatures and validation of input temperatures for each day (with prompts, range check and (nested)iteration) **R2** Calculate, round to one decimal place and store daily average temperatures and calculate the weekly average temperature rounded to one decimal place (iteration, totalling and rounding) **R3** Convert all average temperatures to Fahrenheit (to one decimal place) and output the average temperatures in both Celsius and Fahrenheit. Output with appropriate messages. (output and rounding)	**15**

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

```
// meaningful identifiers and appropriate data structures for
// all data required
DECLARE Days : ARRAY[1:7] OF STRING
DECLARE Readings : ARRAY[1:7, 1:24] OF REAL
DECLARE AverageTemp : ARRAY[1:7] OF REAL
DECLARE WeekLoop : INTEGER
DECLARE DayLoop : INTEGER
DECLARE InTemp : REAL
DECLARE TotalDayTemp : REAL
DECLARE TotalWeekTemp : REAL
DECLARE AverageWeekTemp : REAL
// initial population of Days[] array
// input and a loop are also acceptable
Days[1] ← "Sunday"
Days[2] ← "Monday"
Days[3] ← "Tuesday"
Days[4] ← "Wednesday"
Days[5] ← "Thursday"
Days[6] ← "Friday"
Days[7] ← "Saturday"
// input temperatures inside nested loop
FOR WeekLoop ← 1 TO 7
    TotalDayTemp ← 0
    FOR DayLoop ← 1 TO 24
        OUTPUT "Enter temperature ", DayLoop, " for ", Days[WeekLoop]
```

Question	Answer	Marks

11	INPUT InTemp
	// validation of input for between -20 and +50 inclusive
	WHILE InTemp < -20.0 OR InTemp > 50.0 DO
	OUTPUT "Your temperature must be between -20.0 and +50.0 inclusive. Please try again"
	INPUT InTemp
	ENDWHILE
	Readings[WeekLoop, DayLoop] ← InTemp
	// totalling of temperatures during the day
	TotalDayTemp ← TotalDayTemp + ROUND(InTemp, 1)
	NEXT DayLoop
	
	// average temperature for the day
	AverageTemp[WeekLoop] ← ROUND(TotalDayTemp / 24,1)
	NEXT WeekLoop
	// calculate the average temperature for the week
	TotalWeekTemp ← 0
	FOR WeekLoop ← 1 TO 7
	TotalWeekTemp ← TotalWeekTemp + AverageTemp[WeekLoop]
	NEXT WeekLoop
	
	AverageWeekTemp ← ROUND(TotalWeekTemp / 7,1)
	// outputs in Celsius and Fahrenheit
	FOR WeekLoop ← 1 TO 7
	OUTPUT "The average temperature on ", Days[WeekLoop], " was ", AverageTemp[WeekLoop], " Celsius and ", ROUND(AverageWeekTemp * 9 / 5 + 32), 1, " Fahrenheit"
	NEXT WeekLoop
	
	OUTPUT "The average temperature for the week was ", AverageWeekTemp," Celsius and ", ROUND(AverageWeekTemp * 9 / 5 + 32, 1)," Fahrenheit"	

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 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.*