CAMBRIDGE
International Education

Cambridge IGCSE™ (9–1)

COMPUTER SCIENCE **0984/21**

Paper 2 Algorithms, Programming and Logic **May/June 2024**

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 May/June 2024 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 descriptions 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	**D**	**1**

Question	Answer	Marks
2(a)	**One** mark for each correct line **Description** → **Programming concept** a subroutine that may **not** return a value → procedure a value that is declared and used within a specific procedure → local variable a value that a procedure expects you to supply when it is called → parameter a subroutine that will always return a value → function	**4**
2(b)	**One** mark per mark point • Correct key word - `CALL` • Procedure name with correct parameters - `Average(25, 50)` `CALL Average(25, 50)`	**2**

Question	Answer	Marks
2(c)	**One** mark per mark point, **max three** • Procedures / functions divide the program into smaller manageable segments • … making it more readable / easier to understand / easier to debug • Procedures / functions with meaningful names help to provide documentation for the program / enable/help/provide abstraction • Procedures and functions may be re-used (in the program / in other programs / as part of a library) / run from a single line • Procedures and functions can reduce / eliminate (repeated) code	**3**

Question	Answer	Marks
3	**One** mark for a correct statement about each data type … … **one** mark for a correct example of data for each data type. Example answer **Integer** A whole number [1] Example 27 [1] **Real** A number that contains a fractional part [1] … Example 18.75 [1]	**4**

Question	Answer	Marks

4(a)	**One** mark per mark point
	• Line 01 / `DECLARE Loop : STRING` should be `DECLARE Loop : INTEGER`
	• Line 07 / `IF Loop ← 1 TO Limit` should be `FOR Loop ← 1 TO Limit`
	• Line 09 / `INPUT Loop` should be `INPUT Value`
	• Line 10 / `Total ← Total * Value` should be `Total ← Total + Value`
	Correct algorithm:
	`01 DECLARE Loop : INTEGER`
	`02 DECLARE Limit : INTEGER`
	`03 DECLARE Value : REAL`
	`04 DECLARE Total : REAL`
	`05 Total ← 0`
	`06 Limit ← ROUND(RANDOM() * 19,0) + 1`
	`07 FOR Loop ← 1 TO Limit`
	`08     OUTPUT "Enter a number"`
	`09     INPUT Value`
	`10     Total ← Total + Value`
	`11 NEXT Loop`
	`12 OUTPUT "The total of the numbers entered is ", Total`
	`13 OUTPUT "The average number entered is ", Total / Limit`
		4

Question	Answer	Marks

4(b)	**One** mark per mark point
	• Correct use of `ROUND` with 2 arguments separated by comma, for example ROUND(5, 2) (in statement 13) …
	• … `(Total / Limit,1)` correct arguments
	
	For example:
	
	`OUTPUT "The average of the numbers entered is ",`
	`ROUND(Total / Limit,1)`
		2

4(c)	**One** mark per mark point, **max four** • After line 09 / after the input • Insert a `WHILE` / pre-condition loop… • … to check if the value entered is between 1 and 500 inclusive • If the value is not in range, output an error message • … and insert another input statement for re-input. **Or** • Before line 08 / before the input message • start a `REPEAT` / post-condition loop • After line 09 / after the input • … close the `REPEAT` loop by checking if the value entered was between 1 and 500 inclusive • If it wasn't, (the loop will repeat) for the number to be re-input / If it was, the program continues.	**4**

Question	Answer	Marks

5	**One** mark per mark point
	
	MP1 Correct input statement with appropriate variable
	MP2 Elements of selection statement present – `CASE OF ENDCASE`
	MP3 At least one correct branch in the case statement
	MP4 All branches from 1 to 4 correct
	MP5 Correct use of `OTHERWISE` with correct output.
	
	For example:
	
	`INPUT Number`
	`CASE OF Number`
	`  1 : OUTPUT Number`
	`  2 : OUTPUT Number`
	`  3 : OUTPUT Number`
	`  4 : OUTPUT Number`
	`  OTHERWISE OUTPUT "ERROR"`
	`ENDCASE`
	
	Or
	
	`INPUT Number`
	`CASE OF Number`
	`  1 : OUTPUT 1`
	`  2 : OUTPUT 2`
	`  3 : OUTPUT 3`
	`  4 : OUTPUT 4`
	`  OTHERWISE OUTPUT "ERROR"`
	`ENDCASE`
		5

Question	Answer	Marks
6(a)	**One** mark per mark point **MP1** Correct **`Limit`** column **MP2** Correct **`Count`** column **MP3** Array columns for input stage **MP4** Array columns for sort stage **MP5** Correct **`Flag`** column **MP6** Correct **`Swap`** column **MP7** Correct **`Result`** and **OUTPUT** columns	**7**

		`Numbers`										
`Limit`	**`Count`**	**`[1]`**	**`[2]`**	**`[3]`**	**`[4]`**	**`[5]`**	**`[6]`**	**`[7]`**	**`Flag`**	**`Swap`**	**`Result`**	**OUTPUT**
7	1	47										
	2		**`50`**									
	3			52								
	4				60							
	5					80						
	6						63					
	7							70				
	8								TRUE			
	1								FALSE			
	2											
	3											
	4											

Question	Answer	Marks

6(a)

		Numbers										
Limit	Count	[1]	[2]	[3]	[4]	[5]	[6]	[7]	Flag	Swap	Result	OUTPUT
	5									80		
						63	80		TRUE			
	6									80		
							70	80	TRUE			
	7											
	1								FALSE			
	2											
	3											
	4											
	5											
	6											
	7											
											4	60

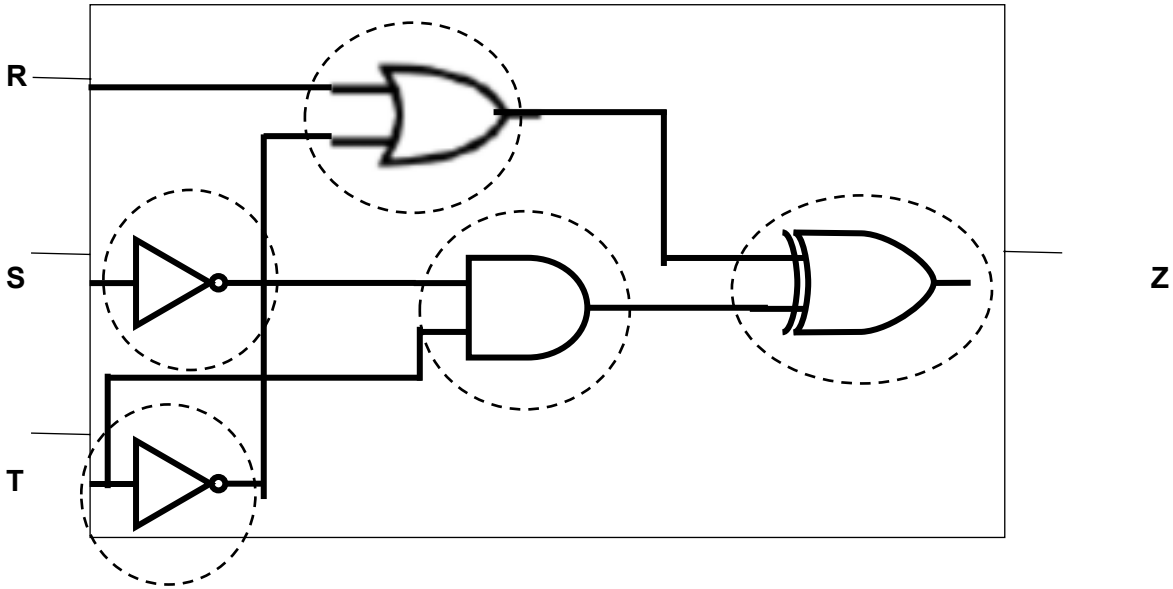
6(b) **3**

One mark per mark point

- A set of numbers is input / stored in an array
- The numbers are sorted / Bubble sort into (ascending) order
- The middle / median value is found / output

Question	Answer	Marks
7(a)	**One** mark per mark point, **max two** • Data stored in a file gives a permanent copy / prevents the data from being lost • … so it can be recalled / used again in a program / in another program • can be stored elsewhere / transferred to another computer	**2**

7(b)	**One** mark per mark point, **max four**
	MP1 Declaration of a string variable
	MP2 Correct use of `OPENFILE` keywords for reading
	MP3 Correct use `READFILE` with string variable
	MP4 Correct use of `UCASE` and `LENGTH` functions **and** output of each
	MP5 Correct use of `CLOSEFILE`
	
	Example:
	
	`DECLARE Words : STRING`
	`OPENFILE Quotation.txt FOR READ`
	`READFILE Quotation.txt, Words`
	`OUTPUT UCASE(Words), LENGTH(Words)`
	`CLOSEFILE Quotation.txt`
		4

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

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>8(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>R</th><th>S</th><th>T</th><th>Z</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</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>1</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>1</td></tr>
</table>
</td><td>4</td></tr>
</table>

Question	Answer	Marks
9(a)	**One** mark for an appropriate reason if no has been stated, for example: None of the fields are likely to contain unique data.	**1**

9(b)	**One** mark for each correct answer
	
	`SUM`
	`NumberInStock`
	`SoftDrinks`
	`Container`
	`'Can'`
	
	Correct code:
	
	`SELECT SUM (NumberInStock)`
	`FROM SoftDrinks`
	`WHERE Container = 'Can';`
		5

10	**Data structures required:** The names underlined must match those given in the scenario: Arrays or lists `Clubs[], Statistics[], Points[], TieIndex[]` Variables `Matches, Won, Drawn, Lost, Counter, Maximum, TieCheck, OutLoop, Max, Count` **Requirements (techniques):** **R1** Input and store number of matches, cricket club names, number of matches won, drawn and lost. Validation of number of matches played and matches won, lost and drawn against number of matches played (with prompts, iteration, storing data in variables, validation, 1D and 2D arrays). **R2** Calculate and store the number of points for each cricket club. Finding the index of the array with the highest score / highest score (calculation, finding the maximum, iteration). **R3** Identifying cricket clubs with highest number of points and whether there is a tie for the top position. Output with appropriate messages (iteration, selection and output).	**15**

Question	Answer	Marks
10	**Example 15-mark answer in pseudocode**	

```
// meaningful identifiers and appropriate data structures
// similar variables grouped to save space as in HL languages
DECLARE Clubs : ARRAY[1:12] OF STRING
DECLARE Statistics : ARRAY[1:12, 1:3] OF INTEGER
DECLARE Points : ARRAY[1 : 12] OF INTEGER
DECLARE TieIndex : ARRAY[1 : 12] OF INTEGER
DECLARE Matches, Won, Drawn, Lost : INTEGER
DECLARE Counter, Maximum, TieCheck, OutLoop : INTEGER
DECLARE Max, Count : INTEGER
// input number of matches
REPEAT
    OUTPUT "How many matches have been played (Maximum 22)? "
    INPUT Matches
UNTIL Matches <= 22
// input of data as a single loop - multiple loops for data entry
// is also acceptable.
FOR Counter ← 1 TO 12
    OUTPUT "Enter the name of the cricket club"
    INPUT Clubs[Counter]
    // input of match results with validation
    REPEAT
        OUTPUT "Enter the number of matches won, drawn and lost for ", Clubs[Counter]
        INPUT Won, Drawn, Lost
        IF Won + Drawn + Lost <> Matches
          THEN
            OUTPUT "Your inputs must total ", Matches, " please try again"
        ENDIF
    UNTIL Matches = Won + Drawn + Lost
    Statistics[Counter, 1] ← Won
    Statistics[Counter, 2] ← Drawn
    Statistics[Counter, 3] ← Lost
    // calculating and storing points
```

Question	Answer	Marks

10	Points[Counter] ← Statistics[Counter, 1] * 12 + Statistics[Counter, 2] * 5
	NEXT Counter
	// finding the highest points
	Max ← -100
	FOR Maximum ← 1 TO 12
	IF Points[Maximum] > Max
	THEN
	Max ← Points[Maximum]
	ENDIF
	NEXT Maximum
	// finding the winner(s)
	Count ← 0
	FOR TieCheck ← 1 TO 12
	IF Points[TieCheck] = Max
	THEN
	Count ← Count + 1
	TieIndex[Count] ← TieCheck
	ENDIF
	Next TieCheck
	// outputting the results
	FOR OutLoop ← 1 TO Count
	OUTPUT "Winning Club(s): ", Clubs[TieIndex[OutLoop]]
	OUTPUT "Number of wins: ", Statistics[TieIndex[OutLoop], 1]
	Next OutLoop
	OUTPUT "Winning Points: ", Max	

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.*