Cambridge IGCSE™ (9–1)

COMPUTER SCIENCE **0984/22**

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

MARK SCHEME

Maximum Mark: 75

Published

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 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.

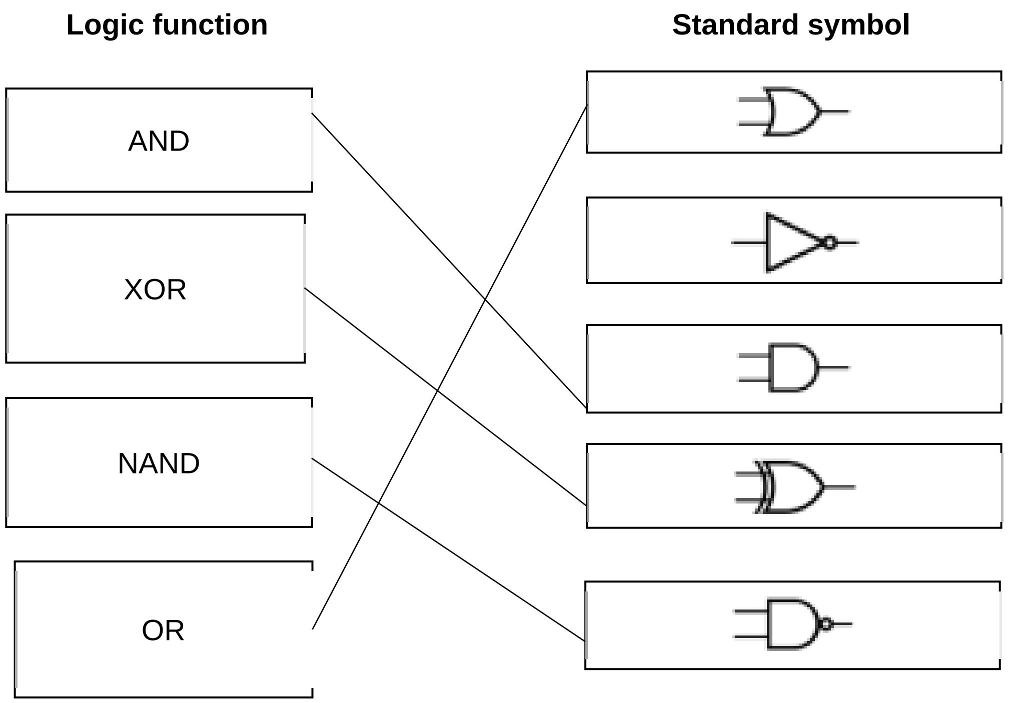
GENERIC MARKING PRINCIPLE 5: Marks should be awarded using the full range of marks defined in the mark scheme for the question (however; the use of the full mark range may be limited according to the quality of the candidate responses seen).
GENERIC MARKING PRINCIPLE 6: Marks awarded are based solely 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
<u>**underline**</u> 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	**C**	**1**

Question	Answer	Marks
2	**One** mark for each correct line  Logic function Standard symbol AND XOR NAND OR	**4**

Question	Answer	Marks
3	**One** mark for each correct answer **max three** MP1 abstraction MP2 decomposition MP3 identification of problem MP4 identification of requirements // outline of success criteria	**3**

Question	Answer	Marks
4(a)	**One** mark per mark point MP1 length check … MP2 … to ensure the product code entered is 6 characters in length MP3 format check … MP4 … to ensure the first two characters of the product code entered are "PD" MP5 range check … MP6 … to ensure that the value of the last four figures of the product code entered is between 1000 and 9999	**6**

4(b)(i)	**One** mark for correct use of `LENGTH` operation, **one** mark for appropriate test
	
	Example:
	
	`REPEAT`
	`   INPUT Product`
	`UNTIL LENGTH(Product) = 6`
		2

4(b)(ii)	**One** mark for correct use of `SUBSTRING` operation, **one** mark for appropriate test
	
	Example:
	
	`REPEAT`
	`   INPUT Product`
	`UNTIL SUBSTRING(Product, 1, 2) = "PD"`
		2

Question	Answer	Marks
5	**One** mark for each description, **one** mark for each example • arithmetic – used in calculations (1) `A ← B + C` (1) • Boolean – used for operations with true or false values (1) `IF B AND C` (1) • logical – used in comparisons/conditional statements/selection statements (1) `IF B > C` (1)	**6**

Question	Answer	Marks
6(a)	**One** mark for: MP1 adding current value to total **One** mark for each point **max three**. MP2 input more than one number MP3 setting total to zero before loop MP4 correct use of loop including terminal condition MP5 output total **after loop** Example: (see below)	**4**

```
Total ← 0
INPUT Value
WHILE Value <> 9999.9
    Total ← Total + Value
    INPUT Value
ENDWHILE
OUTPUT Total

Value ← 0
Total ← 0
REPEAT
    Total ← Total + Value
    INPUT Value
UNTIL Value = 9999.9
OUTPUT Total
```

Question	Answer	Marks

6(b)	**One** mark for each point
	
	MP1 adding one to counter
	MP2 **correct** use of selection, if current value > 100 THEN … ENDIF
	
	One mark for each point, **max two**
	
	MP3 input more than one number
	MP4 setting counter to zero before loop
	MP5 correct use of loop including terminal condition
	MP6 output value of counter **after loop**
	
	Example:
	
	`Counter ← 0`
	`INPUT Value`
	`WHILE Value <> 9999.9`
	`    IF Value > 100`
	`      THEN`
	`        Counter ← Counter + 1`
	`    ENDIF`
	`    INPUT Value`
	`ENDWHILE`
	`OUTPUT Counter`
		4

Question	Answer	Marks

7(a)	`01`//`02`//`06`//`10`
	`04`(`07`) and/or `08`
	`03`(`12`)
		3

Question	Answer	Marks

7(b)	**One** mark for each error identified and corrected
	
	Line `04` `<` should be `>`
	Line `08` `Count` should be `Counter`
	Line `11` `ENDWHILE` should be `ENDIF`
	
	`01 Max ← List[1]`
	`02 Min ← List[1]`
	`03 FOR Counter ← 2 TO 1000`
	`04     IF List[Counter]` **`>`** `Max`
	`05       THEN`
	`06         Max ← List[Counter]`
	`07     ENDIF`
	`08     IF List[`**`Counter`**`] < Min`
	`09       THEN`
	`10         Min ← List[Counter]`
	`11     `**`ENDIF`**
	`12 NEXT Counter`
	`13 OUTPUT "Maximum value is ", Max`
	`14 OUTPUT "Minimum value is ", Min`
		3

Question	Answer	Marks

8(a)	X = 1 **mark**
	(A AND B) // A AND B 1 **mark**
	AND NOT C 1 **mark**
	
	X = (A AND B) AND NOT C
		3

Question	Answer	Marks
8(b)	**Four** marks for 8 correct outputs **Three** marks for 6/7 correct outputs **Two** marks for 4/5 correct outputs **One** mark for 2/3 correct outputs	**4**

A	B	C	X
0	0	0	**0**
0	0	1	**0**
0	1	0	**0**
0	1	1	**0**
1	0	0	**0**
1	0	1	**0**
1	1	0	**1**
1	1	1	**0**

Question	Answer	Marks
9(a)	**One** mark for each of columns **A**, **B** and **T** **Two** marks for columns **List[1]** to **List[5]** all entries correct or **One** mark for columns **List[1]** to **List[5]** with one error	**5**

A	B	List[1]	List[2]	List[3]	List[4]	List[5]	T
		15	17	20	5	9	
FALSE	1	17	15				15
TRUE	2		20	15			15
TRUE	3						
TRUE	4				9	5	5
	5						
FALSE	1	20	17				17
TRUE	2						
	3						
	4						
	5						
FALSE	1						
	2						
	3						
	4						
	5						

Question	Answer	Marks
9(b)	**One** mark for each point MP1 (bubble) sort data in array MP2 in descending order	**2**

Question	Answer	Marks
10(a)	`ContractNumber`	**1**
10(b)	**One** mark for every **two** correct data types **Field** — **Data type** `ContractNumber` — text/alphanumeric `Months` — integer `EndDate` — date/time `Sport` — Boolean	**2**
10(c)	**One** mark for each point MP1 to find the total number of months for **all** contracts MP2 to find the number of contracts MP3 … that are subscribed to `News`	**3**

10(d)	`ContractNumber`
	`News AND Sport` // `Sport AND News`
	
	Example answer:
	`SELECT ContractNumber`
	`FROM Contract`
	`WHERE News // News = TRUE AND Sport // Sport = TRUE ;`
		2

Question	Answer	Marks
11	**Data Structures required** with names as given in the scenario: Arrays or lists <u>`Grid`</u> **Requirements (techniques)** **R1** Set up game – generate random cell, clear all other cells in array, set player start position and start player moves counter (iteration, use of arrays and library routines (round and random)) **R2** Input and check move – is it valid? (input, output, iteration and selection) **R3** Decide outcome – has move found the X? If so, give appropriate output. If not increment counter and continue. If 10 moves exceeded, give appropriate output (use of arrays, iteration, selection and output). ***Example 15-mark answer in pseudocode***	**15**

```
// Set up game
FOR Row ← 1 TO 5
    FOR Column ← 1 TO 5
        Grid[Row, Column] ← ''// set grid cells to be empty
    NEXT Column
NEXT Row
```

Question	Answer	Marks

11	
	```
	REPEAT  // not in cell 1,1
	    XRow ← ROUND ((RANDOM() * 4) + 1, 0) // Random row position between 1 and 5 in GRID
	    XColumn ← ROUND ((RANDOM() * 4) + 1, 0) // Random column position between 1 and 5 in
	GRID
	UNTIL XRow <> 1 and XColumn <> 1 // not in cell 1,1

	Grid [XRow, XColumn] ← 'X'
	MaxMove ← 10
	NumberMoves ← 0
	PlayerRow ← 1
	PlayerColumn ← 1
	Win ← FALSE

	// during game
	WHILE NumberMoves < MaxMove AND NOT Win
	    MoveError ← FALSE
	    OUTPUT "Please enter your move, L – Left, R – Right, U – Up or D - Down"
	    INPUT UPPER(PlayerMove)
	    REPEAT
	        CASE OF PlayerMove
	          'L' : TempColumn ← PlayerColumn – 1
	          'R' : TempColumn ← PlayerColumn + 1
	          'U' : TempRow ← PlayerRow – 1
	          'D' : TempRow ← PlayerRow + 1
	          OTHERWISE MoveError ← TRUE
	        ENDCASE

	// check for out-of-range moves
	        IF TempColumn < 1 or TempColumn > 5
	          THEN
	            MoveError ← TRUE
	          ELSE
	            PlayerColumn ← TempColumn
	        ENDIF
	```

Question	Answer	Marks

11	
	```
	        IF TempRow < 1 or TempRow > 5
	          THEN
	            MoveError ← TRUE
	          ELSE
	            PlayerRow ← TempRow
	        ENDIF

	// check win if X Found
	        IF Grid [PlayerRow, PlayerColumn] = 'X'
	          THEN
	            OUTPUT "You Win"
	            Win ← TRUE
	          ELSE
	            IF NOT MoveError
	              THEN
	                NumberMoves ← NumberMoves + 1
	            ENDIF
	        ENDIF
	    UNTIL NOT MoveError
	ENDWHILE

	IF NOT Win
	  THEN
	    OUTPUT "You Lose"
	ENDIF
	```

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 attempts to meet most of the requirements. *Solution contains lines of code that attempt 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.*