CAMBRIDGE
International Education

Cambridge IGCSE™

COMPUTER SCIENCE **0478/21**

Paper 2 Algorithms, Programming and Logic **October/November 2025**

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 2025 series for most Cambridge IGCSE, Cambridge International A and AS Level components, and some Cambridge O Level components.

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.

Annotations guidance for centres

Examiners use a system of annotations as a shorthand for communicating their marking decisions to one another. Examiners are trained during the standardisation process on how and when to use annotations. The purpose of annotations is to inform the standardisation and monitoring processes and guide the supervising examiners when they are checking the work of examiners within their team. The meaning of annotations and how they are used is specific to each component and is understood by all examiners who mark the component.

We publish annotations in our mark schemes to help centres understand the annotations they may see on copies of scripts. Note that there may not be a direct correlation between the number of annotations on a script and the mark awarded. Similarly, the use of an annotation may not be an indication of the quality of the response.

The annotations listed below were available to examiners marking this component in this series.

Annotations

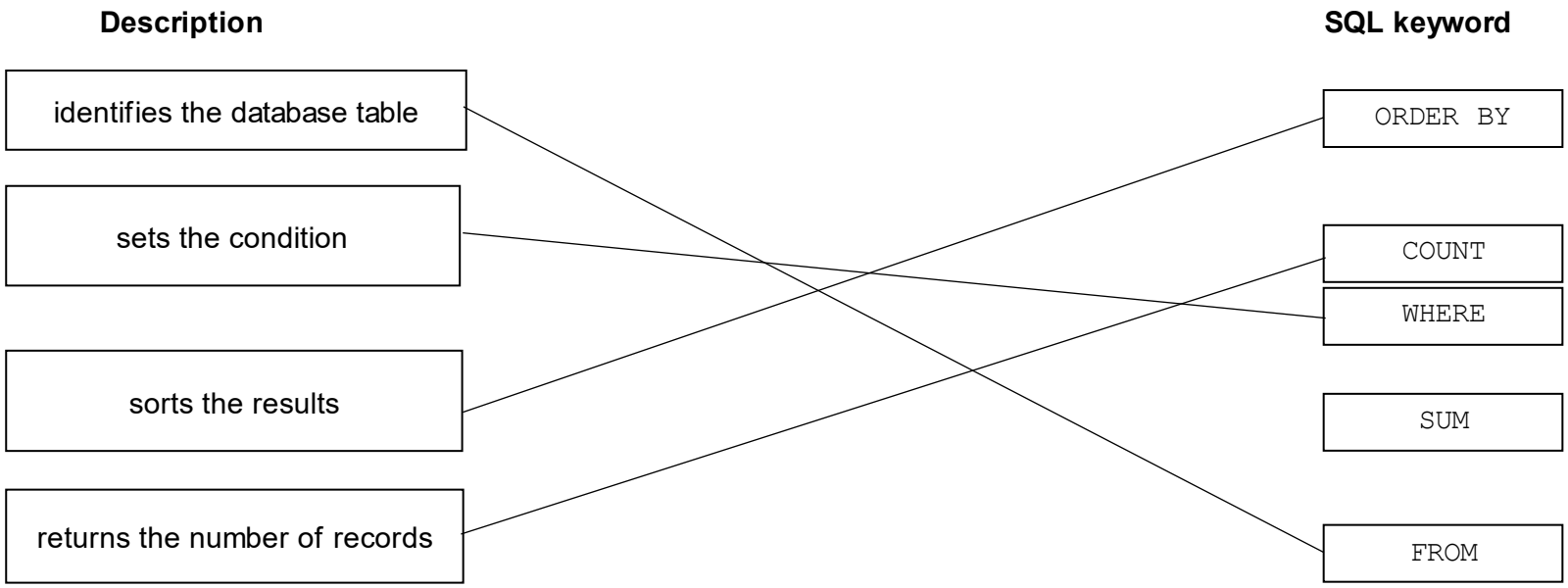
Annotation	Meaning
	Correct point
	Incorrect point
	Follow through
	Repetition
	Ignore
	Benefit of doubt given
	Content of response too vague
	Not answered question
	Omission
	Section not relevant

Annotation	Meaning
~~	Section incorrect
Highlighter	Highlights part of the answer or shows structure of complex answers
SEEN	Page or response seen by examiner
A2	AO2 mark
A3	AO3 mark
NE	Not enough
R1	Required item one
R2	Required item two
R3	Required item three
✓ 1	Correct awarding one mark
✓ 2	Correct awarding two marks
✓ 3	Correct awarding three marks
✓ 4	Correct awarding four marks
✓ 5	Correct awarding five marks
✓ 6	Correct awarding six marks
✓ 7	Correct awarding seven marks
✓ 8	Correct awarding eight marks
✓ 9	Correct awarding nine marks

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	**One** mark for each correct line 	**4**

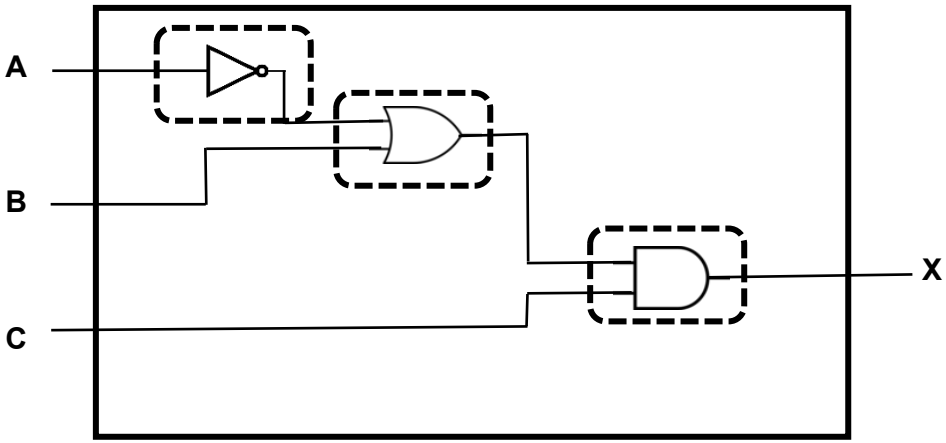
Question	Answer	Marks
2	**One** mark for each correct answer, Max **3** • normal • abnormal • extreme • boundary	**3**

Question	Answer	Marks
3	B	**1**

Question	Answer	Marks

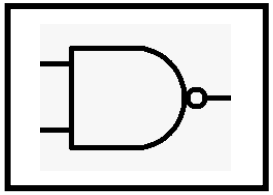
4(a)	**One** mark per bullet point
	
	• input of adult and child number
	• calculation of total without discount
	• check if total people are 6 or more
	• … apply discount to total
	• output of the total
	
	Example:
	
	`OUTPUT "Please enter the number of adults "`
	`INPUT Adult`
	`OUTPUT "Please enter the number of children "`
	`INPUT Child`
	`Total ← (Adult * 9.99) + (Child * 6.99)`
	`IF (Adult + Child) >= 6`
	`  THEN`
	`    Total ← Total – (Total * 15/100)`
	`ENDIF`
	`OUTPUT "The total cost is ", Total`
		5

4(b)	**One** mark per bullet point • Correct use of `ROUND` • … to 2 decimal places Example `Total ← ROUND(Total, 2)`	**2**

Question	Answer	Marks
4(c)(i)	**One** mark per bullet point • **(** NOT **A** OR **B)** • AND **C** **(X** =) (NOT **A** OR **B)** AND **C**	**2**
4(c)(ii)	 **One** mark per correct gate	**3**

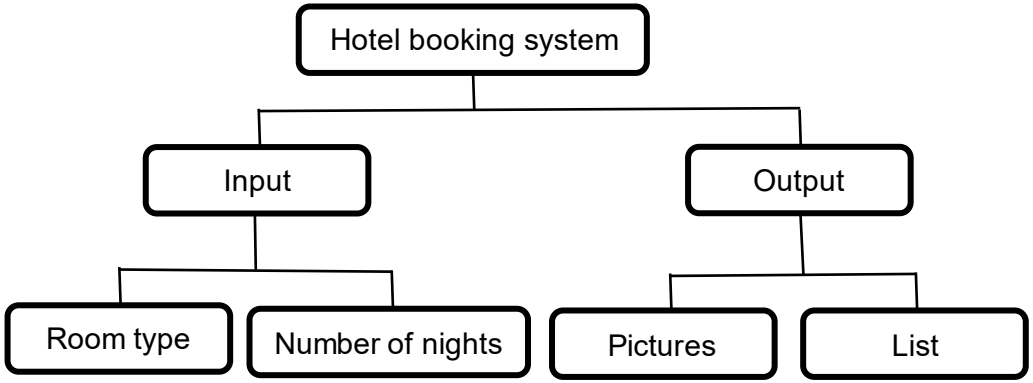
<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>4(d)</td><td>

- **One** mark for correct symbol
- **One** mark for correctly completed table



A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

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

Question	Answer	Marks
5	• **One** mark for a suitable hierarchical structure • **One** mark for suitable names for the sub systems for user input and display options • **One** mark for sub systems for user inputs, room type and number of nights • **One** mark for sub systems for display output types, pictures and list 	**4**

Question	Answer	Marks
6(a)	**One** mark for each line identified and the correction: Line 03 – should be `DECLARE CheckDigit : INTEGER` Line 05 – should be `FOR Count ← 1 TO 5` Line 08 – should be `Total ← Total + (Digit * Count)` Line 15 – should be `OUTPUT "The check digit is ", CheckDigit`	**4**
6(b)	**One** mark per bullet point, Max **3** • After the for loop but before the total/after line 07 • use a suitable loop (repeat/while) • … checking digit is 0 to 9 • re-input digit in the loop	**3**
6(c)	**One** mark for identification and **one** mark for correct description for each: • visual check • … look at the data that has been entered and confirm it matches • double entry • … enter data twice and only accept identical values	**2**

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>7(a)</td><td>One mark per column across both tables:

Input data: 20

<table>
<tr><th>Count</th><th>Number</th><th>Value</th><th>Output</th></tr>
<tr><td>0</td><td>20</td><td>0</td><td></td></tr>
<tr><td>1</td><td></td><td>20</td><td></td></tr>
<tr><td>2</td><td></td><td>40</td><td></td></tr>
<tr><td>3</td><td></td><td>60</td><td></td></tr>
<tr><td>4</td><td></td><td>80</td><td></td></tr>
<tr><td>5</td><td></td><td>100</td><td></td></tr>
<tr><td>6</td><td></td><td>120</td><td></td></tr>
<tr><td>5</td><td></td><td></td><td>5</td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
</table>
</td><td>4</td></tr>
</table>

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>7(a)</td><td>Input data: 110

<table>
<tr><th>Count</th><th>Number</th><th>Value</th><th>Output</th></tr>
<tr><td>0</td><td>110</td><td>0</td><td></td></tr>
<tr><td>1</td><td></td><td>110</td><td></td></tr>
<tr><td>0</td><td></td><td></td><td>0</td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
</table></td><td></td></tr>
<tr><td>7(b)</td><td>One mark from:

• DIV algorithm
• Integer division
• to find how many times the number goes into 100</td><td>1</td></tr>
<tr><td>7(c)</td><td>Integer</td><td>1</td></tr>
</table>

Question	Answer	Marks

7(d)	**One** mark per point:
	
	Either
	• declare Count
	• initialise Count
	• use of DIV
	• … with correct parameters
	
	Or
	• declare Count
	• use of while/repeat
	• … with correct condition
	• output of correct count
	
	Examples:
	
	`DECLARE Count : INTEGER`
	`Count ← 0`
	`INPUT Number`
	`Value ← Count * Number`
	`WHILE Value <= 100 DO`
	`  Count ← Count + 1`
	`  Value ← Count * Number`
	`ENDWHILE`
	`Count ← Count - 1`
	`OUTPUT Count`
	
	Or
	
	`DECLARE Count : INTEGER`
	`(Count ← 0)`
	`INPUT Number`
	`Count ← DIV(100, Number)`
	`OUTPUT Count`
		4

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>8(a)</td><td>One mark from:

• parent name can be repeated
• not unique</td><td>1</td></tr>
<tr><td>8(b)</td><td>StudentID</td><td>1</td></tr>
<tr><td>8(c)</td><td>Two marks for 3 correct data types or one mark for 1 or 2 correct data types

<table><tr><th>Field</th><th>Data type</th></tr><tr><td>ParentName</td><td>Text</td></tr><tr><td>Permission</td><td>Boolean/text</td></tr><tr><td>DateReceived</td><td>Date/time/text</td></tr></table></td><td>2</td></tr>
<tr><td>8(d)</td><td>One mark if two correct, Two marks if all three correct

<pre>SELECT StudentName
FROM SchoolTrip
WHERE Permission = False;</pre></td><td>2</td></tr>
</table>

Question	Answer	Marks

9	**One** mark per bullet point
	• correctly opening `Attendance.txt` for read
	• use of `READFILE`
	• … and storing in `StudentAttendance`
	• closing `Attendance.txt`
	
	Example
	
	`OPENFILE Attendance.txt FOR READ`
	`READFILE Attendance.txt, StudentAttendance`
	`CLOSEFILE Attendance.txt`
		4

Question	Answer	Marks

10(a)	**One** mark for both declarations correct
	
	`DECLARE Seconds : INTEGER`
	`DECLARE Minutes : INTEGER`
		1

10(b)	**One** mark per bullet point
	
	• `PROCEDURE` and `ENDPROCEDURE` naming the procedure `Time` with parameter
	• correctly calculating minutes
	• correctly calculating seconds
	• output the minutes and seconds
	
	Example
	
	`PROCEDURE Time(TotalSeconds : INTEGER)`
	`  Minutes ← DIV(TotalSeconds, 60)`
	`  Seconds ← MOD(TotalSeconds, 60)`
	`  OUTPUT "The number of minutes is ", Minutes`
	`  OUTPUT "The number of seconds is ", Seconds`
	`ENDPROCEDURE`
		4

Question	Answer	Marks
11	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: `CompetitorName`, `CompetitorScore` **Requirements (techniques)** R1 inputs the score for each round validating the input is in the range 0 to 30 inclusive (input, output, nested iteration, validation) R2 calculates the score of each competitor discarding the highest and lowest scores of the 10 rounds (nested iteration, output and selection) R3 outputs the name of the competitor and whether they have qualified, are a reserve or not qualified, output the number of competitors who have qualified, are reserve and not qualified (totalling, output and selection)	**15**

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

```
// programming techniques of iteration, selection, counting, totalling and output are used
// Candidates did not need to declare any arrays or variables.
DECLARE Score : INTEGER
DECLARE Highest : INTEGER
DECLARE Lowest : INTEGER
DECLARE Qualified : INTEGER
DECLARE Reserve : INTEGER
DECLARE NotQualified : INTEGER

Qualified ← 0
Reserve ← 0
NotQualified ← 0

FOR X ← 1 TO 30  //looping for number of competitors
    FOR Round ← 1 TO 10  //looping for number of rounds
        OUTPUT CompetitorName[X]
        OUTPUT "Enter the score for the round: ", Round
        INPUT Score
        // validation of input for between 0 and 30 inclusive
        WHILE Score < 0 OR Score > 30 DO
            OUTPUT "The score must be between 0 and 30 inclusive. Please try again."
            INPUT Score
        ENDWHILE
        CompetitorScore[X, Round] ← Score  // adding score to 2-D array
    NEXT Round
NEXT X


// or suitable method of determining how many competitors
FOR X ← 1 TO 30
    Count ← 0
    Highest ← 0
    Score ← 0
    Lowest ← 999 // resetting variables for each competitor
```

Question	Answer	Marks

11	FOR Y ← 1 TO 10 // loop through rounds
	Score ← Score + CompetitorScore[X, Y]
	IF CompetitorScore[X, Y] > Highest // finds highest score
	THEN
	Highest ← CompetitorScore[X, Y]
	ENDIF
	IF CompetitorScore[X, Y] < Lowest // finds lowest score
	THEN
	Lowest ← CompetitorScore[X, Y]
	ENDIF
	NEXT Y
	Score ← Score – Highest – Lowest // calculating score to subtract highest and lowest
	
	IF Score >= 210 // competitor has qualified
	THEN
	OUTPUT "Competitor ", CompetitorName[X], " has qualified"
	Qualified ← Qualified + 1
	ELSE
	IF Score >= 180 AND Score <= 209 // competitor is a reserve competitor
	THEN
	OUTPUT "Competitor ", CompetitorName[X], " are a reserve"
	Reserve ← Reserve + 1
	ELSE
	OUTPUT "Competitor ", CompetitorName[X], " has not qualified" // competitor not qualified
	NotQualified ← NOTQualified + 1
	ENDIF
	NEXT X
	OUTPUT "The number of competitors qualified is ", Qualified // output of results
	OUTPUT "The number of competitors who are a reserve is ", Reserve
	OUTPUT "The number of competitors not qualified is ", NotQualified	

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>11</td><td>
<table>
<tr><td colspan="4">Marking Instructions in italics</td></tr>
<tr><td 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</td></tr>
<tr><td>0</td><td>1–3</td><td>4–6</td><td>7–9</td></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>
</td><td></td></tr>
</table>

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>11</td><td>
<table>
<tr><td colspan="4">Marking Instructions in italics</td></tr>
<tr><td colspan="4">AO3: Provide solutions to problems by:
• evaluating computer systems
• making reasoned judgements
• presenting conclusions</td></tr>
<tr><td>0</td><td>1–2</td><td>3–4</td><td>5–6</td></tr>
<tr><td rowspan="5">No creditable response.</td><td>Program seen without relevant comments.</td><td>Program seen with some relevant comment(s).</td><td>The program has been fully commented.</td></tr>
<tr><td>Some identifier names used are appropriate.
Some of the data structures used have meaningful names.</td><td>The majority of identifiers used are appropriately named.
Most of the data structures used have meaningful names.</td><td>Suitable identifiers with names meaningful to their purpose have been used throughout.
All of the data structures used have meaningful names.</td></tr>
<tr><td>The solution is illogical.</td><td>The solution contains parts that may be illogical.</td><td>The program is in a logical order.</td></tr>
<tr><td>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.</td><td>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.</td><td>The solution is accurate.
Solution logically performs all the tasks given in the scenario. Ignore minor syntax errors.</td></tr>
<tr><td>The solution attempts at least one of the requirements.
Solution contains lines of code that attempt at least one task given in the scenario.</td><td>The solution meets most of the requirements.
Solution contains lines of code that attempts most tasks given in the scenario.</td><td>The solution meets all the requirements given in the question.
Solution performs all the tasks given in the scenario.</td></tr>
</table>
</td><td></td></tr>
</table>