CAMBRIDGE
International Education

Cambridge IGCSE™

COMPUTER SCIENCE **0478/23**

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 mark point, **max 3** • use of meaningful identifiers • commenting • use of procedures/functions	**3**

Question	Answer	Marks
2	**One** mark for each correct line **Description** / **Programming Technique** adding a set of numbers → totalling displaying the result of a calculation → output executing a set of instructions based on a condition → selection executing the same set of instructions multiple times → iteration counting	**4**

Question	Answer	Marks
3	B	**1**

Question	Answer	Marks

4(a)	**One** mark per point:
	
	• input of adult and child places
	• calculation of total of adults and child places
	• … with booking fee added
	• output of the total using an appropriate message
	
	Example:
	
	`OUTPUT "Please enter the number of adult tickets "`
	`INPUT Adult`
	`OUTPUT "Please enter the number of children "`
	`INPUT Child`
	`Total ← (Adult * 12.99) + (Child * 7.99) + 1.99`
	`OUTPUT "The total cost is ", Total`
		4

4(b)	**One** mark per point: • By adding a while loop//repeat until loop • … with correct condition • With a suitable message • input of adult inside the loop	**4**
4(c)(i)	**One** mark per bullet point: • **A** AND **B** • AND NOT **C** **X** = **A** AND **B** AND NOT **C**	**2**

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

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>5(a)</td><td>One mark for each correct column

<table>
<tr><th>Count</th><th>Number</th><th>Lowest</th><th>Highest</th></tr>
<tr><td></td><td></td><td>900</td><td>1</td></tr>
<tr><td>1</td><td>563</td><td>563</td><td>1</td></tr>
<tr><td>2</td><td>21</td><td>21</td><td>1</td></tr>
<tr><td>3</td><td>376</td><td>21</td><td>376</td></tr>
<tr><td>4</td><td>99</td><td>21</td><td>376</td></tr>
<tr><td>5</td><td>400</td><td>21</td><td>400</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>
<tr><td>5(b)</td><td>One mark for reason and one mark for correction

• the highest score is incorrect / algorithm cannot evaluate highest value if lowest evaluated
• replace <code>ELSE</code> on line 08 with <code>ENDIF</code> and remove <code>ENDIF</code> on line 13 / remove nested IF and have single IF statements for Highest/Lowest</td><td>2</td></tr>
</table>

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>6(a)</td><td>One mark for 2 correct rows, two marks for 3 correct rows

<table><tr><th>Name</th><th>OUTPUT</th></tr><tr><td>Ridwan</td><td>5</td></tr><tr><td>Yan</td><td>−1</td></tr><tr><td>Ella</td><td>3</td></tr></table></td><td>2</td></tr>
<tr><td>6(b)</td><td>(Linear) search</td><td>1</td></tr>
</table>

Question	Answer	Marks

6(c)	**One** mark per point:
	
	• initialising Count **and** Position
	• inputting Name outside of loop
	• use of a suitable loop and condition
	• comparison of Name to array position
	• correct incrementation of Count by 1
	• output of correct position
	
	Example
	
	Repeat loop
	`Position ← -1`
	`Count ← 1`
	`OUTPUT "Please enter a name "`
	`INPUT Name`
	`REPEAT`
	`  IF Name = ClassList[Count]`
	`  THEN`
	`  Position ← Count`
	`  Count ← 6`
	`  ELSE`
	`  Count ← Count + 1`
	`  ENDIF`
	`UNTIL Count > 5`
	`OUTPUT Position`
		6

Question	Answer	Marks

6(c)	While loop
	`Position ← -1`
	`Count ← 1`
	`OUTPUT "Please enter a name "`
	`INPUT Name`
	`WHILE Count <= 5 DO`
	`  IF Name = ClassList[Count]`
	`  THEN`
	`  Position ← Count`
	`  Count ← 6`
	`  ELSE`
	`  Count ← Count + 1`
	`  ENDIF`
	`ENDWHILE`
	`OUTPUT Position`
	
	For loop
	`Position ← -1`
	`(Count ← 1)`
	`OUTPUT "Please enter a name "`
	`INPUT Name`
	`FOR Count ← 1 TO 5`
	`  IF Name = ClassList[Count]`
	`  THEN`
	`  Position ← Count`
	`  ENDIF`
	`NEXT Count`
	`OUTPUT Position`	

Question	Answer	Marks
7(a)	**One** mark for each line identified and corrected Line 02 – should be `DECLARE DenaryNumber : INTEGER` Line 05 – should be `FOR Count ← 8 TO 1 STEP -1` Line 06 – should be `BinaryArray[Count] ← MOD(DenaryNumber,2)`	**3**
7(b)	Any **one** from: • Input • Output • Iteration	**1**
7(c)(i)	Range check	**1**

7(c)(ii)	**One** mark per point:
	
	• Use of a suitable loop
	• ..,suitable condition
	• Correct input of DenaryNumber
	
	Example:
	
	`REPEAT`
	`  INPUT DenaryNumber`
	`UNTIL DenaryNumber >= 1 AND DenaryNumber <= 255`
		3

Question	Answer	Marks
8(a)(i)	EquipmentID	**1**
8(a)(i)	**One** mark per point, **max 1** • Unique field • No repeating data	**1**

Question	Answer	Marks
8(b)	**One** mark per point: • `SELECT Description, Name` • `FROM EquipmentLoan` • `WHERE Returned = "No";`	**3**

Question	Answer	Marks

9(a)	**One** mark for each correct line:
	
	`DECLARE Name : STRING`
	`DECLARE Age : INTEGER`
	`DECLARE Found : BOOLEAN`
		3

9(b)(i)	**One** mark for each bullet point
	
	• using `PROCEDURE StoreData` and `ENDPROCEDURE`
	• defining the parameter correctly
	• correctly opening the file for write
	• correctly storing the name
	• correctly closing the file
	
	Example
	
	`PROCEDURE StoreData(Name : STRING)`
	`  OPENFILE People.txt FOR WRITE`
	`  WRITEFILE People.txt, Name`
	`  CLOSEFILE People.txt`
	`ENDPROCEDURE`
		5

Question	Answer	Marks
9 (b)(ii)	One **mark** per bullet point: • use of call • correct procedure call with 'Suella' `CALL StoreData("Suella")`	**2**
10	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: `NumberGenerated[]` **Requirements (techniques)** R1 input and validates the names of the players (iteration, input, output) R2 generates the random numbers for each player. Calculates and store the number of points per player (iteration, totalling and selection) R3 finds and outputs the name and score of the players in order with a suitable message taking into account a draw(iteration, selection and output)	**15**

Question	Answer	Marks

10	**Example 15-mark answer in pseudocode.**
	
	// programming techniques of iteration, selection, totalling and output are used
	
	`OUTPUT "Please enter the name of player 1 "  // input and validate player 1 with`
	`INPUT Player1                              // presence check`
	`WHILE Player1 = "" DO`
	`    OUTPUT "You must enter a name, try again "`
	`    INPUT Player1`
	`ENDWHILE`
	
	`OUTPUT "Please enter the name of player 2 "  // input and validate player 2`
	`INPUT Player2`
	`WHILE Player2 = "" DO`
	`    OUTPUT "You must enter a name, try again "`
	`    INPUT Player2`
	`ENDWHILE`
	
	`// generating the 100 random numbers for each competitor`
	`FOR X ← 1 TO 2`
	`    FOR Y ← 1 TO 100  // generates 100 random number between 1 and 6 for each player`
	`        NumberGenerated[X, Y] = ROUND(RANDOM() * 5, 0) + 1`
	`    NEXT Y`
	`NEXT X`	

Question	Answer	Marks

10	`//comparing the random numbers`
	`FOR X ← 1 to 100`
	`  IF NumberGenerated[1, X] > NumberGenerated[2, X]`
	`  THEN`
	`  Points[1] ← Points[1] + 2`
	`  ELSE`
	`  IF NumberGenerated[1, X] < NumberGenerated[2, X]`
	`  THEN`
	`  Points[2] ← Points[2] + 2`
	`  ENDIF`
	`  ENDIF`
	`  IF NumberGenerated[1, X] = NumberGenerated[2, X] // if tied (the same)`
	`  THEN`
	`  Points[1] ← Points[1] + 1`
	`  Points[2] ← Points[2] + 1`
	`  ENDIF`
	`NEXT X`	

Question	Answer	Marks

10	`// if player 1 and player 2 tie`
	`// could be a function`
	
	`IF Points[1] = Points[2]`
	`  THEN`
	`    REPEAT`
	`        P1 = ROUND(RANDOM * 6, 0) + 1`
	`        P2 = ROUND(RANDOM * 6, 0) + 1`
	`    UNTIL P1 <> P2`
	`    IF P1 > P2`
	`      THEN`
	`        Points[1] ← Points[1] + 2`
	`      ELSE`
	`        Points[2] ← Points[2] + 2`
	`    ENDIF`
	`ENDIF`
	
	`// determine the overall winner`
	`IF Points[1] > Points[2]`
	`  THEN`
	`    OUTPUT "First is ", Player1`
	`    OUTPUT "With a total score of ", Points[1]`
	`    OUTPUT "Second is ", Player2`
	`    OUTPUT "With a total score of ", Points[2]`
	`  ELSE`
	`    OUTPUT "First is ", Player2`
	`    OUTPUT "With a total score of ", Points[2]`
	`    OUTPUT "Second is ", Player1`
	`    OUTPUT "With a total score of ", Points[1]`
	`  ENDIF`	

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>10</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>10</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="6">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 perform 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>