CAMBRIDGE
International Education

Cambridge IGCSE™

COMPUTER SCIENCE **0478/23**

Paper 2 Algorithms, Programming and Logic **May/June 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 May/June 2025 series for most Cambridge IGCSE, Cambridge International A and AS Level components, and some Cambridge O Level components.

This document consists of **23** 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.

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 standard isation 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

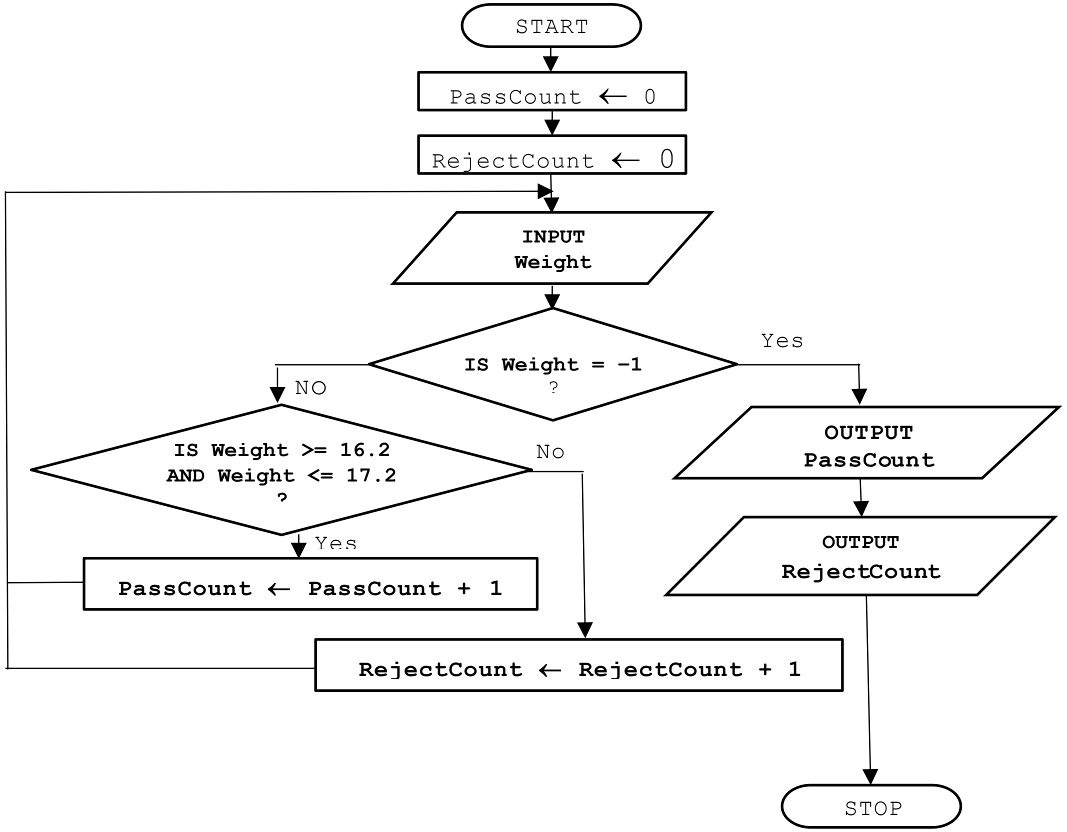
Annotation	Meaning
✓	Correct point
✗	Incorrect point
FT	Follow through
REP	Repetition
I	Ignore
BOD	Benefit of doubt given
TV	Content of response too vague
NAQ	Not answered question
λ	Omission
⌇	Section not relevant

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

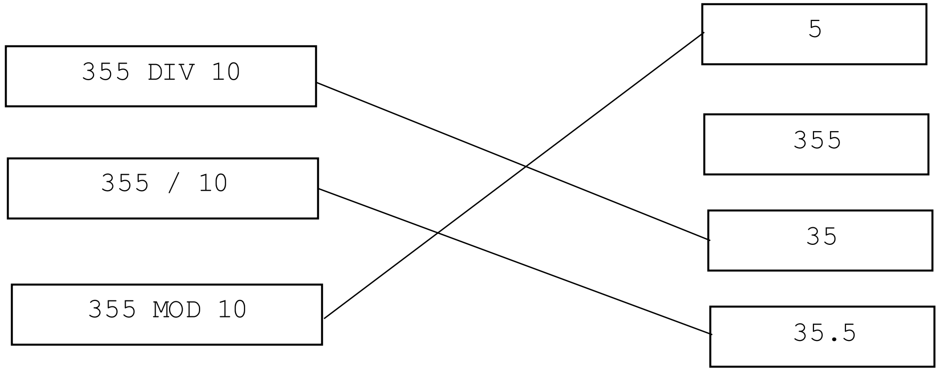
Mark scheme abbreviations

/	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(a)	**One** mark per mark point MP1 input `Weight` MP2 correct check for `Weight = −1` MP3 correct check for `Weight` between 16.2 and 17.2 inclusive MP4 correct increment for `PassCount` MP5 correct increment for `RejectCount` MP6 output `PassCount` and `RejectCount`  START PassCount ← 0 RejectCount ← 0 INPUT Weight IS Weight = −1 ? Yes NO IS Weight >= 16.2 AND Weight <= 17.2 ? No Yes PassCount ← PassCount + 1 RejectCount ← RejectCount + 1 OUTPUT PassCount OUTPUT RejectCount STOP	6

Question	Answer	Marks
1(b)	**One** mark per mark point, **max four** MP1 Initialise a new variable at the start of the flowchart to total the biscuit weights. MP2 Near the point where `PassCount` is incremented/after the yes for the weight decision box, add a new process (box) MP3 … to add the weight of the current biscuit to the running total weight. MP4 Outside the loop, calculate the average weight as total weight divided by number of biscuits that passed (`PassCount`) MP5 Output the average weight, outside the loop.	**4**

Question	Answer	Marks
2	**One** mark for each correct line, **max three** **Use of arithmetic operator** **Result**  355 DIV 10 355 / 10 355 MOD 10 5 355 35 35.5	**3**

Question	Answer	Marks
3(a)	**One** mark per mark point • Line 02 / `DECLARE Index : CHAR` should be `DECLARE Index : INTEGER` • Line 07 / `Stop ← FALSE` should be `Stop ← TRUE` • Line 08 / `FOR Index ← 1 TO 50` should be `FOR Index ← 1 TO 999` // `FOR Index ← 1 TO 1000 - 1` • Line 17 / `NEXT Stop` should be `ENDWHILE`	**4**
3(b)(i)	**One** mark per mark point MP1 Two integer parameters in `PROCEDURE` line MP2 Correct index assigned to `Hold ← Values[Index1]` MP3 Array element correctly assigned to new array element (`Index2` to `Index1`) MP4 Value in `Hold` assigned to correct array element (`Index2`). **For example,** PROCEDURE Swap(**Index1 : INTEGER, Index2 : INTEGER**) DECLARE Hold : REAL Hold ← Values[**Index1**] **Values[Index1] ← Values[Index2]** **Values[Index2] ← Hold** ENDPROCEDURE	**4**

Question	Answer	Marks
3(b)(ii)	**One** mark per mark point MP1 Correct call command and use of `Swap` for procedure name MP2 Two parameters referring to the correct indices of array elements to be swapped, as shown in the original algorithm. `Index + 1` and `Index`. **For example,** `CALL Swap(Index + 1, Index)`	**2**
3(c)	**One** mark per mark point MP1 Global variables can be used throughout the program and its procedures // The memory used by the variables is not recovered until the program terminates. MP2 The variable declared in part 3(b)(i) is a local variable and **can only be used in that procedure** // The memory used by a local variable is **recovered at the end of the procedure**.	**2**

Question	Answer	Marks
4(a)(i)	Range (check)	**1**
4(a)(ii)	**One** mark per correct piece of test data, **max two** **One** mark per correct reason, **max two** **Abnormal test data:** 150 (any non-integer or any value outside the range 10 and 95) **Reason:** This data is out of range so should be **rejected** **Extreme test data**: 10 // 95 **Reason:** This is the smallest data entry that will be **accepted** // This is the largest data entry that will be **accepted**.	**4**
4(b)(i)	Length (check)	**1**

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

One mark per mark point, **max five**

MP1 Appropriate condition-controlled loop
MP2 Password input to a variable
MP3 Use of `LENGTH` function to find password length
MP4 Selection statement to check password length is at least 12
MP5 Error message if length incorrect **inside loop**
MP6 Correct loop termination for password of correct length.

For example,

```
// Variable declarations are not required for this question
REPEAT
    INPUT Password
    IF LENGTH(Password) < 12
      THEN
        OUTPUT "Password too short, please try again"
    ENDIF
UNTIL LENGTH(Password) >= 12
```

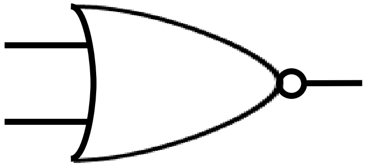
</td><td>5</td></tr>
</table>

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>5(a)</td><td><b>Four</b> marks for all eight data types correct
<b>Three</b> marks for six or seven data types correct
<b>Two</b> marks for four or five data types correct
<b>One</b> mark for two or three data types correct
<table>
<tr><th>Field</th><th>Data type</th><th>Description</th></tr>
<tr><td>ID</td><td><b>text // integer</b></td><td>unique identifier</td></tr>
<tr><td>Name</td><td><b>text</b></td><td>component name</td></tr>
<tr><td>Description</td><td><b>text</b></td><td>component description</td></tr>
<tr><td>Price</td><td><b>real</b></td><td>selling price of component to 2 decimal places</td></tr>
<tr><td>NumberAvailable</td><td><b>integer</b></td><td>number in stock</td></tr>
<tr><td>MinimumLevel</td><td><b>integer</b></td><td>level at which the component is reordered</td></tr>
<tr><td>ReOrdered</td><td><b>Boolean</b></td><td>whether or not the component has been reordered</td></tr>
<tr><td>DateOrdered</td><td><b>date/time</b></td><td>the last date the component was reordered</td></tr>
</table>
</td><td><b>4</b></td></tr>
</table>

Question	Answer	Marks
5(b)	**One** mark per mark point MP1 All correct fields in `SELECT` in any order MP2 Correct table name in `FROM`: `COMPONENT` MP3 Correct field in `WHERE`: `ReOrdered` (=) MP4 Correct criterion in `WHERE`: (=) `TRUE` **Correct code:** `SELECT` **`ID, Name, DateOrdered`** `FROM` **`COMPONENT`** `WHERE` **`ReOrdered = TRUE ;`**	**4**

Question	Answer	Marks
6(a)	**One** mark per mark point MP1 Correct `Word` and `Letter` columns MP2 Correct `Index` column MP3 Correct `Store(1, 3, 4, 6, 8)` columns (consonants) MP4 Correct `Store(2, 5, 7)` columns (vowels) MP5 Correct OUTPUT Column	**5**

			`Store[]`								
`Word`	**`Index`**	**`Letter`**	`[1]`	`[2]`	`[3]`	`[4]`	`[5]`	`[6]`	`[7]`	`[8]`	**OUTPUT**
COMPUTER	1	C	C								
	2	O									
	3	M			M						
	4	P				P					
	5	U									
	6	T						T			
	7	E									
	8	R								R	
	1										C
	2										
	3										M
	4										P
	5										

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>6(a)</td><td>
<table>
<tr><td></td><td>6</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>T</td></tr>
<tr><td></td><td>7</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>8</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>R</td></tr>
</table>
</td><td></td></tr>
<tr><td>6(b)</td><td>
<b>One</b> mark per mark point, <b>max three</b>
MP1 Each letter of a word is checked to see if it is A, E, I, O or U / a vowel
MP2 Vowels are changed to the null string using a case statement
MP3 Each letter or null string is individually stored in an array
MP4 All consonants are output.
</td><td>3</td></tr>
<tr><td>7(a)</td><td>
<b>One</b> mark for the logic gate and <b>one</b> mark for the truth table
<b>NOR Logic gate symbol:</b>

<b>Truth table:</b>
<table>
<tr><th>A</th><th>B</th><th>Z</th></tr>
<tr><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>0</td></tr>
</table>
</td><td>2</td></tr>
</table>

Question	Answer	Marks
7(b)(i)	**One** mark per mark point, **max two** directly from the problem MP1 NOT L AND (M OR S) MP2 OR NOT L AND M AND S **Or** **One** mark per mark point, **max two** with some simplification from the problem MP3 NOT L AND MP4 (M OR S OR (M AND S)) **Or** **One** mark per mark point, **max two** with more simplification from the problem MP5 NOT L AND MP6 (M OR S) **Example answers:** **(A =)** NOT L AND (M OR S) OR NOT L AND M AND S // **(A =)** NOT L AND (M OR S OR (M AND S)) // **(A =)** NOT L AND (M OR S)	**2**

Question	Answer	Marks
7(b)(ii)	**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	**4**

L	M	S	A
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Question	Answer	Marks
8	Marks are available for: • AO2 (maximum 9 marks) • AO3 (maximum 6 marks) **Data Structures required** with names as given in the scenario: Arrays or lists `Video[]`, `Results[]` **Requirements (techniques)** **R1** initialises arrays, displays menu and allows choice, then proceeds based on valid choice, (nested iteration, input, output, selection, validation). **R2** enters data and stores into array at first available location. The code allows for additional data to be input, as required. (counting, selection, input, storage, iteration if selected). **R3** finds specific data based on user input and outputs results. Appropriate output messages, including if video not found. Program continues until stopped by user. (input, output, linear search, iteration).	**15**

Question	Answer	Marks

Example 15-mark answer in pseudocode

```
//Array and variable declarations not required in responses
//Initialisation of Video array
FOR Index1 ← 1 TO 10000
    FOR Index2 ← 1 TO 4
        Video[Index1, Index2] ← ""
    Next Index2
NEXT Index1
//Display of menu choices
REPEAT
    OUTPUT "Enter 1 to input data for a new video, 2 to search for a video title, or 3 to stop"
    //Input menu choice with validation of input
    REPEAT
        INPUT Answer
        IF Answer <> 1 AND Answer <> 2 AND Answer <> 3
          THEN
            OUTPUT "You must input 1, 2 or 3, please try again"
        ENDIF
    UNTIL Answer = 1 OR Answer = 2 OR Answer = 3
    //User chooses to input data for a new video title
    IF Answer = 1
      THEN
        Store ← 1
        //Finding the next available space in the array
        REPEAT
            IF Video[Store, 1] <> ""
              THEN
                 Store ← Store + 1
            ENDIF
        UNTIL Video[Store, 1] = ""
        //Data entry and storage until user says no more needed
        REPEAT
            OUTPUT "Enter the title"
            INPUT Video[Store, 1]
            OUTPUT "Enter the format (4K for 4K, BD for Blu-ray, DV for DVD, DG for Digital"
            INPUT Video[Store, 2]
            OUTPUT "Enter the year of release"
            INPUT Video[Store, 3]
```

Question	Answer	Marks

```
        OUTPUT "Enter the storage code"
        INPUT Video[Store, 4]
        Store ← Store + 1
        OUTPUT "Another video (Y or N)?"
        INPUT Another
    UNTIL Another = 'N' OR Another = 'n'
ENDIF
//User chooses to search for a video title
IF Answer = 2
  THEN
    REPEAT
        //Initialisation of Results array
        FOR Index1 ← 1 TO 20
            FOR Index2 ← 1 TO 4
                Results[Index1, Index2] ← ""
            NEXT Index2
        NEXT Index1
        //User inputs their search criterion
        OUTPUT "State the title of the video you want"
        INPUT Title
        Search ← 1
        Store ← 1
        //Searching for the title
        REPEAT
            //If found, data copied to Results array
            IF Video[Search, 1] = Title
              THEN
                FOR Index1 ← 1 TO 4
                    Results[Store, Index1] ← Video[Search, Index1]
                NEXT Index1
                //Search and Store indexes incremented
                Search ← Search + 1
                Store ← Store + 1
              ELSE
                //Search index incremented
                Search ← Search + 1
            ENDIF
```

Question	Answer	Marks

```
            //Search ends when no more data in Video array,
            //or end of Video array reached.
            UNTIL Video[Search, 1] = "" OR Search = 10000
            //Outputting only results found from Results array
            IF Results[1, 1] <> ""
              THEN
                OUTPUT "The results: "
                 FOR Index1 ← 1 TO Store - 1
                     FOR Index2 ← 1 TO 4
                        OUTPUT Results[Index1, Index2]
                    NEXT Index2
                NEXT Index1
              ELSE
              //Feedback to user if item not found
                OUTPUT "Item not found"
            ENDIF
            //Another search offered
            OUTPUT "Another search (Y or N)?"
            INPUT Another
        UNTIL Another = 'N' OR Another = 'n'
    ENDIF
UNTIL Answer = 3
```

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 attempts 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.