

Cambridge IGCSE™

COMPUTER SCIENCE**0478/23**

Paper 2 Algorithms, Programming and Logic

May/June 2026

MARK SCHEME

Maximum Mark: 75

Published

This mark scheme contains the instructions and marking guidance that our examiners used to mark candidate responses for this component.

Before they start marking, examiners complete a standardisation process. They review a range of candidate responses and discuss the mark scheme in detail to agree which answers can and cannot be accepted. Examiners award marks where it is clear that candidate responses have met the requirements of the mark scheme. These requirements may vary between different components or different versions of the same component, depending on the exact requirements of the question and the candidate responses seen during the standardisation process.

Sometimes, our mark schemes may allow marks to be awarded to responses which contain elements that are factually incorrect or for content not covered by the syllabus. We do this in response to the demands of a specific question to support candidates and make sure that our assessments are fair. However, these allowances should not be used as a guide for teaching and learning.

We cannot discuss the mark scheme with schools, and we recommend that schools read the mark scheme together with the assessment material and the Principal Examiner Report for Teachers.

This document consists of **22** printed pages.

General marking principles

All examiners must apply these marking principles when marking candidate responses.

1	Examiners must award marks for each response in line with: - the specific content defined in the mark scheme - the specific skills defined in the mark scheme or in the level descriptions - the standard of response required as exemplified by the standardisation scripts.
2	Examiners must award whole marks (not half marks, or other fractions).
3	Examiners must award marks positively . This means that: - marks are not deducted for errors or omissions - marks are awarded for correct/valid answers as defined in the mark scheme and also for other valid answers which go beyond the scope of the syllabus and mark scheme referring to your team leader as appropriate - the quality of spelling, punctuation and grammar are judged only when the mark scheme indicates that these features are specifically assessed in the question. However, the meaning of all responses must be unambiguous.
4	Examiners must consistently apply the requirements of the mark scheme and any rules for what to do when candidates have not followed instructions correctly.
5	Examiners must use the full range of marks defined in the mark scheme, unless limited by the quality of the candidate responses seen.
6	Examiners must award marks according to the mark scheme and must not award marks with grade thresholds or grade descriptions 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
	Page or response seen by examiner
	AO2 mark
	AO3 mark
	Not enough
	Required item one
	Required item two
	Required item three
	Correct awarding one mark
	Correct awarding two marks
	Correct awarding three marks
	Correct awarding four marks
	Correct awarding five marks
	Correct awarding six marks
	Correct awarding seven marks
	Correct awarding eight marks
	Correct awarding nine marks

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
MP	mark point

Note: No marks are awarded for using brand names of software packages or hardware

Question	Answer	Marks	Guidance
1	D	1	

Question	Answer	Marks	Guidance
2	C	1	

Question	Answer	Marks	Guidance
3(a)(i)	Normal (test data)	1	
3(a)(ii)	Extreme (test data)	1	accept Boundary
3(b)	One mark per mark point: • A value to instantly verify the accuracy of data • that is calculated from the input • An example, for example the last digit of an ISBN code on a book / bar code / credit card / parity bit	3	

Question	Answer	Marks	Guidance
3(c)	One mark per mark point, max 6: • First input for new password • Checking the length of the new password • Means to re-renter the new password if the length check fails, until it passes • Re-input of new password ... • and check if it matches the first input • Means to re-start, indefinitely, the input process from the beginning if the verification fails • At least two outputs seen, from input prompt / error messages / acceptance message. For example: <pre> Accepted ← FALSE REPEAT REPEAT OUTPUT "Input your new password with at least 10 characters" INPUT Password IF LENGTH(Password) < 10 THEN OUTPUT "Your password must contain at least 10 characters, please re-input" ENDIF UNTIL LENGTH(Password) >= 10 OUTPUT "Please re-enter your new password for verification" INPUT Password2 IF Password2 = Password THEN Accepted ← TRUE OUTPUT "Your new password is accepted" ELSE OUTPUT "Your password has not been accepted, please try again" ENDIF UNTIL Accepted </pre>	6	No array or variable declarations required for this algorithm Ignore case

Question	Answer	Marks	Guidance
4(a)	One mark per mark point: - Line 05 / DECLARE RunTime : STRING should be DECLARE RunTime : INTEGER - Line 11 / OUTPUT RunTime should be INPUT RunTime - Line 13 / RunSecs $\leftarrow$ DIV(RunTime, 60) should be RunSecs $\leftarrow$ MOD(RunTime, 60) // RunTime - RunMins * 60 - Line 16 / Runners[Index] $\leftarrow$ RunMins should be Runners [Index] $\leftarrow$ RunName	4	Ignore case
	<pre> 01 DECLARE Runners : ARRAY[1:250] OF STRING 02 DECLARE Times : ARRAY[1:250] OF INTEGER 03 DECLARE Index : INTEGER 04 DECLARE RunName : STRING 05 DECLARE RunTime : INTEGER 06 DECLARE RunMins : INTEGER 07 DECLARE RunSecs : INTEGER 08 FOR Index $\leftarrow$ 1 TO 250 09 OUTPUT "Enter a runner's name and then time(in seconds) " 10 INPUT RunName 11 INPUT RunTime 12 RunMins $\leftarrow$ DIV(RunTime, 60) 13 RunSecs $\leftarrow$ MOD(RunTime, 60) 14 OUTPUT "Runner's Name: ", RunName 15 OUTPUT "Time: ", RunMins, " Minutes ", RunSecs, " Seconds" 16 Runners[Index] $\leftarrow$ RunName 17 Times[Index] $\leftarrow$ RunTime 18 NEXT Index </pre>		

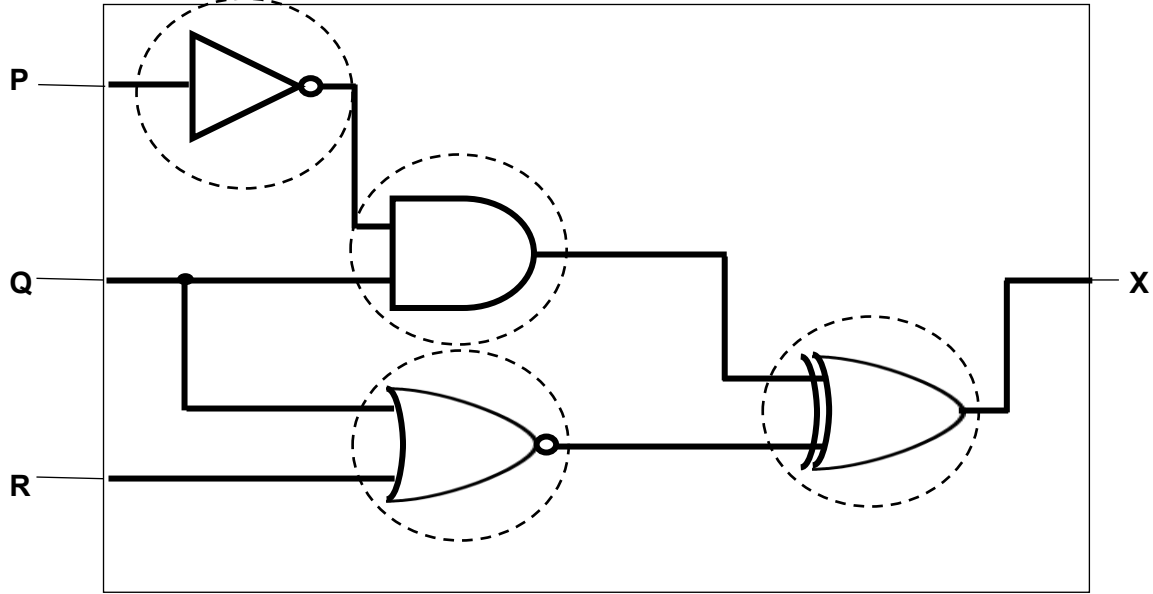
Question	Answer	Marks	Guidance
4(b)(i)	Any five from: • Use a condition-controlled loop inside the existing loop before the <code>RunTime</code> input • <code>REPEAT // WHILE RunTime <1 OR RunTime > 3600 (DO)</code> • After line 11 / <code>INPUT RunTime, ...</code> • insert a selection statement with a range check (and error message) • <code>IF RunTime < 1 OR RunTime > 3600 (not required for WHILE loop version)</code> • <code>OUTPUT "The time must be positive and no greater than 3600 seconds"</code> • Finish the repeat loop before the calculations start / before line 12 • <code>UNTIL RunTime >0 AND RunTime <= 3600 // ENDWHILE</code>	5	Ignore case BOD use of 0 as a valid input for MP 2, MP5 and MP8

Question	Answer	Marks	Guidance
4(b)(i)	Example: <pre> 01 DECLARE Runners : ARRAY[1:250] OF STRING 02 DECLARE Times : ARRAY[1:250] OF INTEGER 03 DECLARE Index : INTEGER 04 DECLARE RunName : STRING 05 DECLARE RunTime : INTEGER 06 DECLARE RunMins : INTEGER 07 DECLARE RunSecs : INTEGER 08 FOR Index ← 1 TO 250 09 OUTPUT "Please enter a runner's name and time in seconds " 10 INPUT RunName REPEAT 11 INPUT RunTime IF RunTime < 1 OR RunTime > 3600 THEN OUTPUT "The time must be positive and not greater than 3600 seconds" ENDIF UNTIL RunTime >0 AND RunTime <= 3600 12 RunMins ← DIV(RunTime, 60) 13 RunSecs ← MOD(RunTime, 60) 14 OUTPUT "Runner's Name: ", RunName 15 OUTPUT "Time: ", RunMins, " Minutes ", RunSecs, "Seconds" 16 Runners[Index] ← RunName 17 Times[Index] ← RunTime 18 NEXT Index </pre>		
4(b)(ii)	One mark from: Example: • The original FOR loop will always iterate 250 times, even if there are not 250 runners. • Only 250 runners are allowed in the array. • Two athletes could have the same name, so would be difficult to identify. • Answers related to validation or verification are acceptable.	1	

Question	Answer				Marks	Guidance	
5(a)	One mark for each correct column.				5	If no other marks awarded, allow 1 mark for correct initialisation of PassCount, FailCount and Count to 0 Repeated errors such as not initialising all columns, penalise once and FT Ignore case in output, but punctuation must be correct	
	PassCount	FailCount	Count	Size			OUTPUT
	0	0	0				
	1		1	49.8			
	2		2	48.7			
		1	3	62.5			
	3		4	49.9			
	4		5	30.0			
		2	6	58.2			
	5		7	−12.5			
		3	8	50.0			
	6		9	26.7			
							Number passed: 6
							Number failed: 3

Question	Answer	Marks	Guidance
5(b)	Any two from: - Add another decision box after the input box and before the existing decision box. - Check for values <0 inside this decision box - If the condition fails, loop back to the input box for re-input // if the condition passes, go to the next decision box	2	This is a flowchart question, so answers must be in the context of a flowchart, or changes to the existing flowchart must be identifiable. Accept answers written as flowchart boxes.

Question	Answer	Marks	Guidance
6(a)	One mark per mark point: - Correct use of <code>LENGTH</code> function with <code>Fact</code> - Correct use of <code>UCASE</code> function with <code>Fact</code> - Contents of <code>LENGTH</code> and <code>UCASE</code> both output (can be both in same <code>OUTPUT</code> statement) Example: <pre>Fact ← "I learnt to program in computer science" OUTPUT LENGTH(Fact) OUTPUT UCASE(Fact)</pre>	3	Ignore case in keywords as pseudocode Case in any strings must match the question paper.
6(b)	One mark per mark point: - Correct use of <code>SUBSTRING</code> with <code>Fact</code> - and two other parameters (13 and 7) as an <code>OUTPUT</code> statement, accept 12 and 7 Example: <pre>OUTPUT SUBSTRING(Fact, 13, 7, accept OUTPUT SUBSTRING(Fact, 12, 7)</pre>	2	

Question	Answer	Marks	Guidance
7(a)	One mark for each correct gate, with the correct input(s) as shown. 	4	

Question	Answer	Marks	Guidance																																				
7(b)	Four marks for eight correct outputs (X column) 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 border="1"> <thead> <tr> <th>P</th><th>Q</th><th>R</th><th>X</th></tr> </thead> <tbody> <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>0</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>1</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>0</td></tr> <tr><td>1</td><td>1</td><td>1</td><td>0</td></tr> </tbody> </table>	P	Q	R	X	0	0	0	1	0	0	1	0	0	1	0	1	0	1	1	1	1	0	0	1	1	0	1	0	1	1	0	0	1	1	1	0	4	
P	Q	R	X																																				
0	0	0	1																																				
0	0	1	0																																				
0	1	0	1																																				
0	1	1	1																																				
1	0	0	1																																				
1	0	1	0																																				
1	1	0	0																																				
1	1	1	0																																				

Question	Answer	Marks	Guidance
8(a)	One mark per mark point: - primary key: <code>SubjectCode</code> - Reason: it is unique for that subject and level	2	

Question	Answer	Marks	Guidance
8(b)	One mark per mark point: • <code>PracticalExams/TheoryExams/MinimumClassSize/MaximumClassSize/CourseLength/TheoryExams</code> • must be represented by whole numbers • <code>CourseWork</code> • this is a yes/no field, either there is coursework or there is no coursework	4	
8(c)	One mark per mark point: • At least two correct fields in <code>SELECT</code> • Remaining correct field in <code>SELECT</code> • Correct <code>FROM OPTION_SUBJECTS</code> • Correct <code>WHERE MinimumClassSize < 10</code> • Correct <code>ORDER BY SubjectCode;</code> Correct code: <pre>SELECT SubjectCode, SubjectName, Level FROM OPTION_SUBJECTS WHERE MinimumClassSize < 10 ORDER BY SubjectCode (ASC);</pre>	5	Ignore case on field names, but spelling must match names in table. If space in field/table name, penalise once and FT. The default ordering is ascending, so award the mark if ASC is there or not. Penalise if anything else for example if DESC or ALPHABETICAL is included instead.

Question	Answer	Marks	Guidance
9	Max six marks from any two categories: One mark per mark point • Use of commenting • to describe sections of code and their function/purpose • to make the code easier for other programmers to understand // to make specific parts of the code easier to find One mark per mark point (Max 3) • Use of meaningful identifier (names) • for variables / constants / procedures / arrays • so that the purpose is suggested by the identifier / name • so that specific program sections / parts are easier to find / identify One mark per mark point (Max 3) • Use of procedures/functions/subprograms • so that the overall program is modular • so that repeated code is avoided • the program is easier to read • the program is shorter // fewer lines of code • the subprogram can be called anywhere it is needed (with different parameters) One mark per mark point • Use of whitespace/indentation • so that the program is laid out clearly/ is more readable • so that the beginning and end of groups of code, such as selection statements/iteration are clear	6	Max two per technique if the technique is not named.

Question	Answer	Marks	Guidance
10	Use the tables for AO2 and AO3 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 with names as given in the scenario: Arrays or lists <code>RiverNames[]</code>, <code>RiverStats[]</code>, <code>DaysCount[]</code> Requirements (techniques) R1 Initialise <code>RiverStats</code> array, input and store the names of the rivers in the array, input daily BOD for 30 days and store it in the array (initialisation, iteration, input, output). R2 Calculate the average daily reading for each river rounded to 2 d.p.. Count the number of moderately polluted days and the number of highly polluted days for each river (totalling, counting, iteration, rounding, selection). R3 Output the average daily BOD readings, the number of moderately polluted days and the number of highly polluted days for each river. Identify and output the most polluted river (output, selection, finding max).	15	

Question	Answer	Marks	Guidance
10	Example 15-mark answer in pseudocode <pre>// note that array and variable declarations are not required in candidate responses DECLARE RiverStats : ARRAY[1:30, 1:5] OF REAL DECLARE DaysCount : ARRAY[1:5, 1:2] OF INTEGER DECLARE RiverNames : Array[1:5] OF STRING DECLARE Index1, Index2 : Integer DECLARE TotalBOD : REAL DECLARE AverageBOD : REAL DECLARE MostPollution : INTEGER DECLARE MostPollRiver : String // initialising array FOR Index1 ← 1 TO 30 FOR Index2 ← 1 TO 5 RiverStats[Index1, Index2] ← 0 NEXT Index2 NEXT Index1 // input river names FOR Index1 ← 1 TO 5 OUTPUT "Input the name of river ", Index1 INPUT RiverNames[Index1] NEXT Index1 // input of daily BOD readings FOR Index1 ← 1 TO 30 OUTPUT "Day ", Index1 FOR Index2 ← 1 TO 5 OUTPUT "Input the reading for ", RiverNames[Index2] INPUT RiverStats[Index1, Index2] NEXT Index2 NEXT Index1 // finding the average daily BOD and counting the number of moderately and highly polluted days FOR Index1 ← 1 TO 5</pre>		

Question	Answer	Marks	Guidance
10	<pre> TotalBOD ← 0 DaysCount[Index1, 1] ← 0 DaysCount[Index1, 2] ← 0 FOR Index2 ← 1 TO 30 TotalBOD ← TotalBOD + RiverStats[Index2, Index1] IF RiverStats[Index2, Index1] >= 2 AND RiverStats[Index2, Index1] <= 8 THEN DaysCount[Index1, 1] ← DaysCount[Index1, 1] + 1 ENDIF IF RiverStats[Index2, Index1] > 8 THEN DaysCount[Index1, 2] ← DaysCount[Index1, 2] + 1 ENDIF NEXT Index2 // calculating and rounding the average BOD to 2 dp AverageBOD ← ROUND(TotalBOD / 30, 2) OUTPUT RiverNames[Index1], " has an average daily pollution rate of ", AverageBOD OUTPUT "with ", DaysCount[Index1, 1], " moderately polluted day(s)" OUTPUT "and ", DaysCount[Index1, 2], " highly polluted day(s)" OUTPUT "" NEXT Index1 // finding and outputting the most polluted river MostPollRiver ← RiverNames[1] MostPollution ← DaysCount[1, 2] </pre>		

Question	Answer	Marks	Guidance
10	<pre> FOR Index1 ← 2 TO 5 IF DaysCount[Index1, 2] > MostPollution THEN MostPollution ← DaysCount[Index1, 2] MostPollRiver ← RiverNames[Index1] ENDIF NEXT Index1 OUTPUT "The most polluted river is ", MostPollRiver OUTPUT "It had ", MostPollution, " day(s) in the last 30 days with a high pollution reading" </pre>		

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. <i>Any use of selection, iteration, counting, totalling, input and output.</i>	Some programming techniques used are appropriate to the problem. <i>More than one technique seen applied to the scenario, check the list of techniques needed.</i>	The range of programming techniques used is appropriate to the problem. <i>All criteria stated for the scenario have been covered by the use of appropriate programming techniques, check the list of techniques needed.</i>
	Some data has been stored but not appropriately. <i>Any use of variables or arrays or other language dependent data structures e.g. Python lists.</i>	Some of the data structures chosen are appropriate and store some of the data required. <i>More than one data structure used to store data required by the scenario.</i>	The data structures chosen are appropriate and store all the data required. <i>The data structures used store all the data required by the scenario.</i>

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. <i>Some of the data structures used have meaningful names.</i>	The majority of identifiers used are appropriately named. <i>Most of the data structures used have meaningful names.</i>	Suitable identifiers with names meaningful to their purpose have been used throughout. <i>All the data structures used have meaningful names.</i>
	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. <i>Solution contains few lines of code with errors that attempt to perform a task given in the scenario.</i>	The solution contains parts that are inaccurate. <i>Solution contains lines of code with some errors that logically perform tasks given in the scenario. Ignore minor syntax errors.</i>	The solution is accurate. <i>Solution logically performs all the tasks given in the scenario. Ignore minor syntax errors.</i>
	The solution attempts at least one of the requirements. <i>Solution contains lines of code that attempt at least one task given in the scenario.</i>	The solution attempts to meet most of the requirements. <i>Solution contains lines of code that attempt most tasks given in the scenario.</i>	The solution meets all the requirements given in the question. <i>Solution performs all the tasks given in the scenario.</i>