



# Cambridge IGCSE™ (9–1)

---

**COMPUTER SCIENCE**

**0984/02**

Paper 2 Algorithms, Programming and Logic

**For examination from 2023**

MARK SCHEME

Maximum Mark: 75

---

**Specimen**

---

This document has **16** pages. Any blank pages are indicated.

**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 descriptors 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
- 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

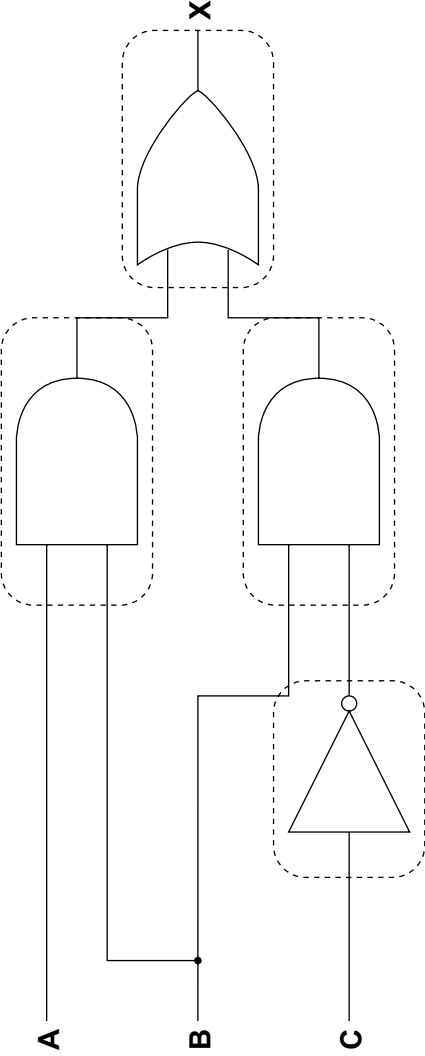
**Note:** No marks are awarded for using brand names of software packages or hardware.

Question	Answer	Marks
<p>1(a)</p> <p><b>Pseudocode description</b></p> <p>a loop that will always iterate at least once</p> <p>a conditional statement to deal with many possible outcomes</p> <p>a loop that will always iterate a set number of times</p> <p>a conditional statement with different outcomes for true and false</p>	<p><b>Pseudocode statement</b></p> <p>FOR...TO...NEXT</p> <p>IF...THEN...ELSE...ENDIF</p> <p>WHILE...DO...ENDWHILE</p> <p>CASE...OF...OTHERWISE...ENDCASE</p> <p>REPEAT...UNTIL</p>	4

Question	Answer	Marks
1(b)	<ul style="list-style-type: none"> <li>• appropriate loop controls</li> <li>• read from array</li> <li>• output from array (the last two points can be in one statement, see example)</li> </ul> <p><b>Note:</b> reading and the output <b>MUST</b> be within the same loop.</p> <p>For example:  Count ← 0  WHILE Count &lt; 50 DO    OUTPUT Name [Count]    Count ← Count + 1  ENDWHILE</p>	3

Question	Answer	Marks
2	<p><b>Validation check</b></p> <p><b>One mark for description:</b></p> <ul style="list-style-type: none"> <li>• To test if the data entered is possible / reasonable / sensible.</li> <li>• A range check tests that data entered fits within specified values.</li> </ul> <p><b>One mark for example:</b></p> <ul style="list-style-type: none"> <li>• Allow any correct validation check as an example (range, length, type, presence, format, etc.).</li> </ul> <p><b>Verification check</b></p> <p><b>One mark for description:</b></p> <ul style="list-style-type: none"> <li>• To test if the data input is the same as the data that was intended to be input.</li> <li>• A double entry check expects each item of data to be entered twice and compares both entries to check they are the same.</li> </ul> <p><b>One mark for example:</b>  Allow any correct verification check as an example (visual, double entry, etc.).</p>	4

Question	Answer	Marks
3	<b>B</b>	<b>1</b>
Question	Answer	Marks
4	<p>One mark for a hierarchical structure.            One mark for suitable names for the sub-systems.            One mark for identifiable inputs.            One mark for identifiable outputs.</p> <p>For example:</p> <pre> graph TD     A[Satellite navigation system] --&gt; B[Input destination]     A --&gt; C[Output directions]     B --&gt; D[New destination]     B --&gt; E[Saved destination]     C --&gt; F[Map]     C --&gt; G[List]           </pre>	<b>4</b>
Question	Answer	Marks
5(a)	<p><b>One mark for each error identified and correction:</b></p> <ul style="list-style-type: none"> <li>Numbers should be <code>Number</code></li> <li><code>IF Number &gt; 100</code> should be <code>IF Number &gt;= 100</code></li> <li><code>INPUT Number</code> is missing from inside the loop, insert <code>INPUT Number</code> after the <code>ENDIF</code> statement.</li> <li>The final <code>OUTPUT Number</code> is not needed, remove it.</li> </ul>	<b>4</b>
5(b)	<p><b>One mark for both ends of the range and correct inequality symbols.</b>  <b>One mark for the <code>AND</code> // nested IFs.</b>            The test should be <code>IF Number &gt;= 100 AND Number &lt;= 200</code></p>	<b>2</b>

Question	Answer	Marks																																				
6(a)	<p>One mark for each correct gate, with the correct input(s) as shown.</p> 	4																																				
6(b)	<table border="1" data-bbox="719 1485 1187 1939"> <thead> <tr> <th>A</th> <th>B</th> <th>C</th> <th>X</th> </tr> </thead> <tbody> <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>1</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>1</td> </tr> </tbody> </table> <p>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.</p>	A	B	C	X	0	0	0	0	0	0	1	0	0	1	0	1	0	1	1	0	1	0	0	0	1	0	1	0	1	1	0	1	1	1	1	1	4
A	B	C	X																																			
0	0	0	0																																			
0	0	1	0																																			
0	1	0	1																																			
0	1	1	0																																			
1	0	0	0																																			
1	0	1	0																																			
1	1	0	1																																			
1	1	1	1																																			

Question	Answer										Marks	
7(a)	<b>Flag</b>	<b>Count</b>	<b>Name [1]</b>	<b>Name [2]</b>	<b>Name [3]</b>	<b>Name [4]</b>	<b>Temp</b>					<b>5</b>
			Jamal	Amir	Eve	Tara						
	0	1	Amir	Jamal	Eve	Tara	Jamal					
	1	2	Amir	Eve	Jamal	Tara	Jamal					
	1	3	Amir	Eve	Jamal	Tara	Jamal					
	1	4	Amir	Eve	Jamal	Tara	Jamal					
	0	1	Amir	Eve	Jamal	Tara	Jamal					
	0	2	Amir	Eve	Jamal	Tara	Jamal					
	0	3	Amir	Eve	Jamal	Tara	Jamal					
	0	4	Amir	Eve	Jamal	Tara	Jamal					
	<p><b>One mark for Flag column.</b>  <b>One mark for Count column.</b>  <b>One mark for Temp column.</b>  <b>Two marks for all correct Name columns or one mark for two or three correct Name columns.</b></p> <p><b>Note:</b> Repeated values do not need to be written unless a value is rewritten.</p> <ul style="list-style-type: none"> <li>bubble sort / sorting the names</li> <li>ascending order / A to Z / lowest to highest / alphabetical order</li> </ul>										<b>2</b>	
7(b)											<b>2</b>	
Question	Answer										Marks	
8	<p><b>One mark for each.</b></p> <p>10.00 boundary / abnormal data // the price should be rejected // value is out of range            9.99 boundary / extreme / normal data // the price should be accepted // value is within normal range            ten abnormal data // input should be rejected // value is wrong type</p>										<b>3</b>	



Question	Answer	Marks
9	<p>Any <b>three</b> from:</p> <ul style="list-style-type: none"> <li>data is not lost when the computer is switched off // data is stored permanently</li> <li>data can be used by more than one program or reused when a program is run again</li> <li>data can be backed up or archived</li> <li>data can be transported from one place / system to another.</li> </ul>	<b>3</b>

Question	Answer	Marks
10	<b>C</b>	<b>1</b>

Question	Answer	Marks										
11(a)	20	<b>1</b>										
11(b)(i)	CatNo	<b>1</b>										
11(b)(ii)	it is a unique identifier	<b>1</b>										
11(c)	<p><b>Two</b> marks for four correct answers. <b>One</b> mark for two or three correct answers.</p> <table border="1"> <thead> <tr> <th>Field</th> <th>Data type</th> </tr> </thead> <tbody> <tr> <td>CatNo</td> <td>Text</td> </tr> <tr> <td>Title</td> <td>Text</td> </tr> <tr> <td>Genre1</td> <td>Text</td> </tr> <tr> <td>Streaming</td> <td>Boolean / Text</td> </tr> </tbody> </table>	Field	Data type	CatNo	Text	Title	Text	Genre1	Text	Streaming	Boolean / Text	<b>2</b>
Field	Data type											
CatNo	Text											
Title	Text											
Genre1	Text											
Streaming	Boolean / Text											
11(d)	FROM "Comedy"	<b>2</b>										

Question	Answer	Marks
12(a)	<p>One mark for each correct line.</p> <pre> DEclare X : STRING DEclare Y : INTEGER DEclare Z : INTEGER </pre>	3
12(b)	<p>One mark for storing string in X.  One mark for calling the function length.  One mark for using the correct parameter X.  One mark for using the substring function.  One mark for correct parameters.  One mark for outputting length and substring return values.</p> <p>For example:</p> <pre> X ← "Programming is fun" OUTPUT Length(X) Y ← 16 Z ← 3 OUTPUT SubString(X,Y,Z) </pre>	6

Question	Answer	Marks
13	<p>Read the whole answer, award a mark from both of the following tables and add up the total.</p> <p>Marks are available for:</p> <ul style="list-style-type: none"> <li>• AO2 (maximum 9 marks)</li> <li>• AO3 (maximum 6 marks).</li> </ul> <p>The techniques and the data structures required are listed below. The requirements may be met using a suitable built-in function from the programming language used (e.g. Python, VB.NET or Java).</p> <p><b>Techniques required:</b></p> <p><b>R1</b> Calculate total mark for each student (iteration and totalling).  <b>R2</b> Calculate average mark for each student rounded to the nearest whole number.  <b>R3</b> Selection of grade for each student (selection).  <b>R4</b> Output for each student name, total mark, average mark, grade awarded (output with appropriate messages).  <b>R5</b> Calculate, store and output the number of distinctions, merits, passes and fails for the whole class (iteration, counting and output with appropriate messages).</p> <p><b>Data structures required:</b>  The names underlined must be used as provided in the scenario.</p> <p>Arrays or lists    <u>StudentName</u>, <u>StudentMark</u>,  (TotalMark and AverageMark may be seen but no requirement to store)</p> <p>Variables        <u>ClassSize</u>, <u>SubjectNo</u>, <u>SubjectCounter</u>, <u>StudentCounter</u>  <u>DistinctionNo</u>, <u>MeritNo</u>, <u>PassNo</u>, <u>FailNo</u> could be an array or list</p> <p>Constants        <u>Distinction</u>, <u>Merit</u>, <u>Pass</u> could be variables</p>	15

Question	Answer	Marks
13	<p><b>Example 15 mark answer in pseudocode.</b></p> <pre> // meaningful identifier names and appropriate data structures (variables, constants and the // given arrays) to store all the data required DECLARE TotalMark : ARRAY[1:50] OF INTEGER DECLARE AverageMark : ARRAY[1:50] OF INTEGER DECLARE SubjectCounter : INTEGER DECLARE StudentCounter : INTEGER DECLARE DistinctionNo : INTEGER DECLARE MeritNo : INTEGER DECLARE PassNo : INTEGER DECLARE FailNo : INTEGER  CONSTANT Distinction = 70 CONSTANT Merit = 55 CONSTANT Pass = 40  // initialisation processes for this scenario, initialising the running totals used for // grades and combined totals DistinctionNo ← 0 MeritNo ← 0 PassNo ← 0 FailNo ← 0  FOR StudentCounter ← 1 to ClassSize     TotalMark[StudentCounter] ← 0 NEXT StudentCounter  // programming techniques of iteration, selection, totalling, counting and output are used </pre>	

Question	Answer	Marks
13	<pre> FOR StudentCounter ← 1 to ClassSize FOR SubjectCounter ← 1 to SubjectNo   TotalMark[StudentCounter] ← TotalMark[StudentCounter] + StudentMark[StudentCounter,   SubjectCounter] NEXT SubjectCounter AverageMark[StudentCounter] ← INT((TotalMark[StudentCounter] / SubjectNo) + 0.5) OUTPUT "Name ", StudentName[StudentCounter] OUTPUT "Combined total mark ", TotalMark[StudentCounter] OUTPUT "Average mark ", AverageMark[StudentCounter] IF AverageMark[StudentCounter] &gt;= Distinction THEN   DistinctionNo ← DistinctionNo + 1   OUTPUT "Grade Distinction" ELSE   IF AverageMark[StudentCounter] &gt;= Merit   THEN     MeritNo ← MeritNo + 1     OUTPUT "Grade Merit"   ELSE     IF AverageMark[StudentCounter] &gt;= Pass     THEN       PassNo ← PassNo + 1       OUTPUT "Grade Pass"     ELSE       FailNo ← FailNo + 1       OUTPUT "Grade Fail"     ENDIF   ENDIF ENDIF NEXT StudentCounter  OUTPUT "Number of Distinctions ", DistinctionNo OUTPUT "Number of Merits ", MeritNo OUTPUT "Number of Passes ", PassNo OUTPUT "Number of Fails ", FailNo </pre>	

<b>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</b>			
<b>0</b>	<b>1–3</b>	<b>4–6</b>	<b>7–9</b>
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, refer to 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, refer to 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 that is 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 that is required by the scenario.</i>

<b>AO3: Provide solutions to problems by:</b> <ul style="list-style-type: none"> <li>• evaluating computer systems</li> <li>• making reasoned judgements</li> <li>• presenting conclusions</li> </ul>	<b>0</b>	<b>1–2</b>	<b>3–4</b>	<b>5–6</b>				
	Program seen without relevant comments. Some identifier names used are appropriate. <i>Some of the data structures used have meaningful names.</i>	Program seen with some relevant comment(s). The majority of identifiers used are appropriately named. <i>Most of the data structures used have meaningful names.</i>	The program has been fully commented. 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 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 may be illogical. 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. The program is in a logical order. The solution is accurate. <i>Solution logically performs all the tasks given in the scenario. Ignore minor syntax errors.</i>	The solution meets most of the requirements. <i>Solution performs all the 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>
<b>No creditable response</b>								

**BLANK PAGE**