



## Cambridge IGCSE™ (9–1)

---

COMPUTER SCIENCE

0984/22

Paper 2

October/November 2020

MARK SCHEME

Maximum Mark: 50

---

**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 2020 series for most Cambridge IGCSE™, Cambridge International A and AS Level and Cambridge Pre-U components, and some Cambridge O Level components.

---

This document consists of **9** 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 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.

Question	Answer	Marks
1(a)(i)	<p>Any meaningful name for an array related to <b>Task 1</b> – <b>one</b> mark e.g.  <code>SysStore</code>  <code>SysPrice</code></p> <p>Correct purpose related to <b>Task 1</b> – <b>one</b> mark e.g.            ...to store the system (components) that have been purchased            ...to store the (total) price of the system (being purchased)</p>	<b>2</b>
1(a)(ii)	<p>Any meaningful name for a variable related to <b>Task 2</b> – <b>one</b> mark e.g.  <code>Component</code>  <code>TotalPrice</code></p> <p>Correct purpose related to <b>Task 2</b> - <b>one</b> mark e.g.            ... to allow input of a component code            ... to store/calculate the running total price of the system</p>	<b>2</b>
1(a)(iii)	<p>Any meaningful name for a constant related to <b>Task 3</b> – <b>one</b> mark e.g.  <code>Offer5</code>  <code>Offer10</code></p> <p>Correct purpose related to <b>Task 3</b> - <b>one</b> mark e.g.            ... to store the one option discount rate            ... to store the two-option discount rate</p>	<b>2</b>
1(b)	<p>Mark as <b>either</b>:            Two distinct different points  <b>OR</b>            One point and an expansion</p> <p>Example answers:            Real data can be used in calculations directly (which is required of the Price data) (1)            Data can be stored with decimal places (1)</p> <p>Real numbers can be used in calculations (1) which is not possible with strings (1)</p>	<b>2</b>

Question	Answer	Marks
1(c)	<p>Any <b>six</b> from:</p> <p>MP1 At least one input (case, RAM, HDD)  MP2 All three inputs fully prompted  MP3 An attempt at validation of input  MP4 One complete validation of input with error message  MP5 Finding the price for one chosen item  MP6 Finding the prices of the other two chosen items correctly  MP7 Calculation of price of the chosen items  MP8 ...add the basic components cost to the cost of the chosen items  MP9 Storage of chosen items  MP10 Output to show chosen items and price of the computer (with appropriate message)</p> <p>Example answer:</p> <pre> OUTPUT "Which type of Case would you like? Input the Item Code" ComponentFlag ← False WHILE ComponentFlag = False   INPUT CaseCode   Count ← 0   WHILE Count&lt;2 DO     IF CaseCode = ComponentCode [Count]       THEN         CaseIndex ← Count         ComponentFlag ← True         Count ← 2       ENDIF     Count ← Count + 1   ENDWHILE   IF ComponentFlag = False     THEN       OUTPUT "Your case Item Code doesn't exist, please try again"     ENDIF   ENDWHILE OUTPUT "Which type of RAM would you like? Input the Item Code" ComponentFlag ← False WHILE ComponentFlag = False   INPUT RAMCode   Count ← 2   WHILE Count&lt;5 DO     IF RAMCode = ComponentCode [Count]       THEN         RAMIndex ← Count         ComponentFlag ← True         Count ← 5       ENDIF     Count ← Count + 1   ENDWHILE </pre>	6


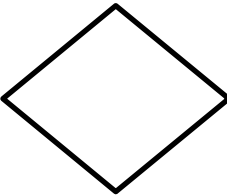
Question	Answer	Marks
1(c)	<pre> IF ComponentFlag = False   THEN     OUTPUT "Your RAM Item Code doesn't exist, please try again"   ENDIF ENDWHILE OUTPUT "Which type of Primary Hard Disk Drive would you like? Input the Item Code" ComponentFlag ← False WHILE ComponentFlag = False   INPUT PHDDCode   Count ← 5   WHILE Count&lt;8 DO     IF PHDDCode = ComponentCode [Count]       THEN         HDDIndex ← Count         ComponentFlag ← True         Count ← 8       ENDIF     Count ← Count + 1   ENDWHILE   IF ComponentFlag = False     THEN       OUTPUT "Your Primary HDD Item Code doesn't exist, please try again"     ENDIF   ENDWHILE TotalPrice ← 200 + ComponentPrice [CaseIndex] + ComponentPrice [RAMIndex] + ComponentPrice [HDDIndex] OUTPUT "Your computer consists of ", Description [CaseIndex], " case, ", Description [RAMIndex], " RAM and ", Description [HDDIndex], " Primary Hard Disk Drive." OUTPUT "The total price of your computer is \$", TotalPrice </pre>	
1(d)	<p><b>Any four</b> from:</p> <p>MP1 Explanation of how the number of additional parts is stored</p> <p>MP2 Explanation of counting of additional parts being added to the system</p> <p>MP3 Explanation of determination of additional parts being 1, or more than 1</p> <p>MP4 Explanation of using the correct percentage discount</p> <p>MP5 Explanation of calculating the money saved and finding the new price</p> <p>MP6 Explanation of correct output of money saved and new price</p>	<b>4</b>
1(e)	<p><b>Any two</b> from:</p> <p>MP1 Prompt and input to ask buyer how many computers they wish to purchase (at the start) // When the first computer is complete, prompt and input to ask if they would like to purchase another computer</p> <p>MP2 Introduce an appropriate loop structure</p> <p>MP3 New storage for more than one computer // Enable the ordering of multiple computers of the same specification</p>	<b>2</b>

Question	Answer	Marks															
<b>Section B</b>																	
2	<table border="1"> <thead> <tr> <th>Statement</th> <th>true (✓)</th> <th>false (✓)</th> </tr> </thead> <tbody> <tr> <td>A subroutine is called from within a program.</td> <td>✓</td> <td></td> </tr> <tr> <td>A subroutine is <b>not</b> a complete program.</td> <td>✓</td> <td></td> </tr> <tr> <td>A subroutine is a self-contained piece of code.</td> <td>✓</td> <td></td> </tr> <tr> <td>A subroutine must return a value to the code from which it was called.</td> <td></td> <td>✓</td> </tr> </tbody> </table> <p><b>Two</b> marks for four correct rows <b>One</b> mark for any two correct rows</p>	Statement	true (✓)	false (✓)	A subroutine is called from within a program.	✓		A subroutine is <b>not</b> a complete program.	✓		A subroutine is a self-contained piece of code.	✓		A subroutine must return a value to the code from which it was called.		✓	2
Statement	true (✓)	false (✓)															
A subroutine is called from within a program.	✓																
A subroutine is <b>not</b> a complete program.	✓																
A subroutine is a self-contained piece of code.	✓																
A subroutine must return a value to the code from which it was called.		✓															

Question	Answer	Marks
3	<p><b>One</b> mark for each correct type of test and <b>one</b> mark for each correct accompanying example of test data and reason (max <b>six</b>) e.g.</p> <ul style="list-style-type: none"> <li>• Extreme data</li> <li>• 5000</li> <li>• to check it is accepted</li>   <li>• Normal data</li> <li>• 300</li> <li>• To check it is accepted</li>   <li>• Abnormal data</li> <li>• 10000</li> <li>• To check it is rejected</li> </ul>	6

Question	Answer	Marks
4	<p>Any <b>six</b> from:</p> <p>MP1 Initialisation of <code>Higher</code> to 0 before the loop</p> <p>MP2 Use of IF statement</p> <p>MP3 Correct condition in IF statement</p> <p>MP4 Correct counting statement inside loop</p> <p>MP5 OUTPUT/PRINT statement with correct reference to <code>Higher</code></p> <p>MP6 Appropriate message in output</p> <p>MP7 Correct location of OUTPUT and IF statements</p> <pre> Higher ← 0 FOR Count ← 1 TO 5000   INPUT Number[Count]   IF Number[Count] &gt; 500     THEN       Higher ← Higher + 1   ENDIF NEXT Count OUTPUT "There are ", Higher, " values that are greater than 500" </pre>	6

Question	Answer								Marks
5(a)	Flag	Count	Num [0]	Num [1]	Num [2]	Num [3]	Num [4]	Store	<b>5</b>
			45	56	30	12	15		
	0	0						45	
			56						
	1			45					
		1							
		2							
		3						12	
						15			
							12		
	0	0							
		1							
		2							
		3							
	<p><b>One mark</b> – Flag column  <b>One mark</b> – Count column  <b>One mark</b> – Num [0] and Num [1] columns  <b>One mark</b> – Num [2], Num [3] and Num [4] columns  <b>One mark</b> – Store column</p>								
5(b)	<p>Any <b>two</b> from:</p> <ul style="list-style-type: none"> <li>• The algorithm sorts/orders numbers</li> <li>• ... into descending order / from largest to smallest</li> </ul>								<b>2</b>

Question	Answer		Marks
6	<p><b>Input/Output</b></p> 	<p><b>Decision</b></p> 	<b>2</b>
<p><b>One mark</b> for each correct symbol</p>			



Question	Answer	Marks																														
7(a)	17	1																														
7(b)	<p><b>One</b> mark for correct fieldname and <b>one</b> mark for correct reason</p> <p>PartNum</p> <p>The data stored in this field is <b>unique</b> for each record</p>	2																														
7(c)	<table border="1" data-bbox="438 510 1337 904"> <tbody> <tr> <td data-bbox="438 510 663 575">Field:</td> <td data-bbox="663 510 888 575">PartNum</td> <td data-bbox="888 510 1114 575">Description</td> <td data-bbox="1114 510 1337 575">Cost</td> <td data-bbox="1337 510 1481 575">Quantity</td> </tr> <tr> <td data-bbox="438 575 663 640">Table:</td> <td data-bbox="663 575 888 640">AUDIOPARTS</td> <td data-bbox="888 575 1114 640">AUDIOPARTS</td> <td data-bbox="1114 575 1337 640">AUDIOPARTS</td> <td data-bbox="1337 575 1481 640">AUDIOPARTS</td> </tr> <tr> <td data-bbox="438 640 663 705">Sort:</td> <td data-bbox="663 640 888 705"></td> <td data-bbox="888 640 1114 705"></td> <td data-bbox="1114 640 1337 705">Descending</td> <td data-bbox="1337 640 1481 705"></td> </tr> <tr> <td data-bbox="438 705 663 770">Show:</td> <td data-bbox="663 705 888 770"><input checked="" type="checkbox"/></td> <td data-bbox="888 705 1114 770"><input checked="" type="checkbox"/></td> <td data-bbox="1114 705 1337 770"><input checked="" type="checkbox"/></td> <td data-bbox="1337 705 1481 770"><input checked="" type="checkbox"/></td> </tr> <tr> <td data-bbox="438 770 663 835">Criteria:</td> <td data-bbox="663 770 888 835"></td> <td data-bbox="888 770 1114 835"></td> <td data-bbox="1114 770 1337 835"></td> <td data-bbox="1337 770 1481 835">&lt;10</td> </tr> <tr> <td data-bbox="438 835 663 904">or:</td> <td data-bbox="663 835 888 904"></td> <td data-bbox="888 835 1114 904"></td> <td data-bbox="1114 835 1337 904"></td> <td data-bbox="1337 835 1481 904"></td> </tr> </tbody> </table> <p data-bbox="296 943 831 1070"> <b>One</b> mark for correct field and table rows  <b>One</b> mark for sort row  <b>One</b> mark for show row  <b>One</b> mark for correct criteria         </p>	Field:	PartNum	Description	Cost	Quantity	Table:	AUDIOPARTS	AUDIOPARTS	AUDIOPARTS	AUDIOPARTS	Sort:			Descending		Show:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Criteria:				<10	or:					4
Field:	PartNum	Description	Cost	Quantity																												
Table:	AUDIOPARTS	AUDIOPARTS	AUDIOPARTS	AUDIOPARTS																												
Sort:			Descending																													
Show:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>																												
Criteria:				<10																												
or:																																