Cambridge Assessment International Education

Cambridge IGCSE™

COMPUTER SCIENCE **0478/22**

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.

© UCLES 2020

[Turn over

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.

© UCLES 2020

Question	Answer	Marks

1(a)(i)	Any meaningful name for an array related to **Task 1** – **one** mark e.g.
	`SysStore`
	`SysPrice`
	
	Correct purpose related to **Task 1** – **one** mark e.g.
	…to store the system (components) that have been purchased
	…to store the (total) price of the system (being purchased)
		2

1(a)(ii)	Any meaningful name for a variable related to **Task 2** – **one** mark e.g.
	`Component`
	`TotalPrice`
	
	Correct purpose related to **Task 2** - **one** mark e.g.
	… to allow input of a component code
	… to store/calculate the running total price of the system
		2

1(a)(iii)	Any meaningful name for a constant related to **Task 3** – **one** mark e.g.
	`Offer5`
	`Offer10`
	
	Correct purpose related to **Task 3** - **one** mark e.g.
	… to store the one option discount rate
	… to store the two-option discount rate
		2

1(b)	Mark as **either**: Two distinct different points **OR** One point and an expansion 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) Real numbers can be used in calculations (1) which is not possible with strings (1)	**2**

© UCLES 2020

Question	Answer	Marks

1(c)	Any **six** from:
	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)
	
	Example answer:
	`OUTPUT "Which type of Case would you like? Input the Item Code"`
	`ComponentFlag ← False`
	`WHILE ComponentFlag = False`
	`  INPUT CaseCode`
	`  Count ← 0`
	`  WHILE Count<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<5 DO`
	`  IF RAMCode = ComponentCode[Count]`
	`  THEN`
	`  RAMIndex ← Count`
	`  ComponentFlag ← True`
	`  Count ← 5`
	`  ENDIF`
	`  Count ← Count + 1`
	`  ENDWHILE`
		6

© UCLES 2020

Question	Answer	Marks

1(c)	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<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	

1(d)	Any **four** from: MP1 Explanation of how the number of additional parts is stored MP2 Explanation of counting of additional parts being added to the system MP3 Explanation of determination of additional parts being 1, or more than 1 MP4 Explanation of using the correct percentage discount MP5 Explanation of calculating the money saved and finding the new price MP6 Explanation of correct output of money saved and new price	**4**
1(e)	Any **two** from: 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 MP2 Introduce an appropriate loop structure MP3 New storage for more than one computer // Enable the ordering of multiple computers of the same specification	**2**

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td colspan="3">Section B</td></tr>
<tr><td>2</td><td>
<table>
<tr><th>Statement</th><th>true (✓)</th><th>false (✓)</th></tr>
<tr><td>A subroutine is called from within a program.</td><td>✓</td><td></td></tr>
<tr><td>A subroutine is not 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>
</table>
Two marks for four correct rows
One mark for any two correct rows
</td><td>2</td></tr>
</table>

Question	Answer	Marks
3	**One** mark for each correct type of test and **one** mark for each correct accompanying example of test data and reason (max **six**) e.g. • Extreme data • 5000 • to check it is accepted • Normal data • 300 • To check it is accepted • Abnormal data • 10000 • To check it is rejected	**6**

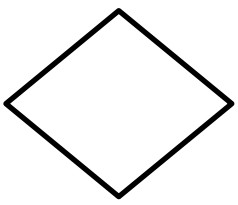
© UCLES 2020

Question	Answer	Marks

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

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>5(a)</td><td>
<table>
<tr><th>Flag</th><th>Count</th><th>Num [0]</th><th>Num [1]</th><th>Num [2]</th><th>Num [3]</th><th>Num [4]</th><th>Store</th></tr>
<tr><td></td><td></td><td>45</td><td>56</td><td>30</td><td>12</td><td>15</td><td></td></tr>
<tr><td>0</td><td>0</td><td></td><td></td><td></td><td></td><td></td><td>45</td></tr>
<tr><td></td><td></td><td>56</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>1</td><td></td><td></td><td>45</td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>1</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>2</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>3</td><td></td><td></td><td></td><td></td><td></td><td>12</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td>15</td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td>12</td><td></td></tr>
<tr><td>0</td><td>0</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>1</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>2</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>3</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>
One mark – <code>Flag</code> column
One mark – <code>Count</code> column
One mark – <code>Num[0]</code> and <code>Num[1]</code> columns
One mark – <code>Num[2], Num[3]</code> and <code>Num[4]</code> columns
One mark – <code>Store</code> column
</td><td>5</td></tr>
<tr><td>5(b)</td><td>Any two from:
• The algorithm sorts/orders numbers
• … into descending order / from largest to smallest</td><td>2</td></tr>
</table>

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>6</td><td>Input/Output

Decision

One mark for each correct symbol</td><td>2</td></tr>
</table>

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>7(a)</td><td>17</td><td>1</td></tr>
<tr><td>7(b)</td><td>One mark for correct fieldname and one mark for correct reason

PartNum

The data stored in this field is unique for each record</td><td>2</td></tr>
<tr><td>7(c)</td><td>
<table>
<tr><td>Field:</td><td>PartNum</td><td>Description</td><td>Cost</td><td>Quantity</td></tr>
<tr><td>Table:</td><td>AUDIOPARTS</td><td>AUDIOPARTS</td><td>AUDIOPARTS</td><td>AUDIOPARTS</td></tr>
<tr><td>Sort:</td><td></td><td></td><td>Descending</td><td></td></tr>
<tr><td>Show:</td><td>☑</td><td>☑</td><td>☑</td><td>☑</td></tr>
<tr><td>Criteria:</td><td></td><td></td><td></td><td><10</td></tr>
<tr><td>or:</td><td></td><td></td><td></td><td></td></tr>
</table>
One mark for correct field and table rows
One mark for sort row
One mark for show row
One mark for correct criteria</td><td>4</td></tr>
</table>

© UCLES 2020