Cambridge Assessment International Education
Cambridge Ordinary Level

COMPUTER SCIENCE **2210/21**

Paper 2 **May/June 2018**

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 May/June 2018 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 **11** 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
Section A		
1(a)(i)	Many correct answers, they must be meaningful. The following is an example only: **One** mark per bullet point Data structure Array Name `processor` Data type string Use to store processors currently available	**4**
1(a)(ii)	**One** mark per bullet point Data structure given (1) Data type (1) Sample data (1) More than one data structure described (1) Many correct answers, they must be meaningful. The following is an example only: e.g. Three arrays containing string data with name, address and phone number – John Smith, Cambridge, 01223 123456	**4**
1(b)	**One** mark for method, **one** mark for an extension or reason. Many correct answers, an example is given. Use a previously stored number//generates/uses an initial value (1) Update it (by 1) every time an estimate is made (1)	**2**

Question	Answer	Marks
1(c)	Any **five** from: 1 Initialise (stock level) flag 2 Check stock level for the chosen processor type 3 Only check RAM if processor available // Only check processor if RAM available 4 Check stock level for the chosen type of RAM 5 Finish process if problem with (RAM/Processor) stock levels 6 Identify out of stock (processor/RAM)//Set flag to appropriate value 7 Identify stock level OK//Set flag to appropriate value	**5**

Question	Answer	Marks
1(c)	Sample answer:	

```
foundProc ← FALSE
count ← 1
WHILE NOT foundProc AND count <=3 DO
  IF processor(estNo) = proc(count) AND stProc(count) > 0
    THEN
      foundProc ← TRUE
  ENDIF
  count ← count + 1
ENDWHILE
IF foundProc
  THEN
    foundRAM ← FALSE
    IF RAM(estNO) = RAM1 AND stRAM1 >0
      THEN
        foundRAM ← TRUE
        stRAM1 ← stRAM1 - 1
    ENDIF
    IF RAM(estNO) = RAM2 AND stRAM2 >0
      THEN
        foundRAM ← TRUE
        stRAM2 ← stRAM2 - 1
    ENDIF
ENDIF
IF NOT foundProc
  THEN
    OUTPUT "Processor out of stock"
  ELSE
    stProc(count) ← stProc(count) - 1
ENDIF
IF NOT foundRAM
  THEN
    OUTPUT "RAM out of stock"
ENDIF
```

Question	Answer	Marks
1(d)	**One** mark for each correct point (max 5): Explanation 1 How the number of orders was calculated 2 Deal with the case where the estimate has not been turned into an order 3 Calculating the total number of each component sold 4 Details of method actually used to calculate numbers of components 5 How the total value of all the orders was calculated 6 Display summary 7 Display complete summary of number of orders, total number of components and total value of orders Programming statements can be used but **must be explained** to gain credit.	**5**

Question	Answer	Marks
	Section B	

2(a)	Any **six** from:
	1 Initialisation of counters for positive numbers and zeros
	2 Appropriate loop for 1000 iterations
	3 Input number inside loop
	4 Test for positive numbers
	5 Update positive number counter
	6 Test for zeros
	7 Update zero counter
	8 Output counters with appropriate messages outside loop
	
	`zero ← 0`
	`posCount ← 0`
	`FOR count ← 1 TO 1000`
	`  INPUT number`
	`  IF number > 0`
	`    THEN posCount ← posCount + 1`
	`  ENDIF`
	`  IF number = 0`
	`    THEN zero ← zero + 1`
	`  ENDIF`
	`NEXT`
	`OUTPUT posCount, " positive numbers"`
	`OUTPUT zero, " zeros"`
		6

2(b)	Reduce the number of iterations to a manageable amount // Simulate the input (e.g. random generation)	**1**

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>3(a)</td><td>
<table>
<tr><th>Digit(1)</th><th>Digit(2)</th><th>Digit(3)</th><th>Digit(4)</th><th>Digit(5)</th><th>Digit(6)</th><th>Digit(7)</th><th>Digit(8)</th><th>Sum</th><th>OUTPUT</th></tr>
<tr><td>5</td><td>7</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>6</td><td>44</td><td>GTIN-8</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>57012346</td></tr>
</table>
<table>
<tr><th>Digit(1)</th><th>Digit(2)</th><th>Digit(3)</th><th>Digit(4)</th><th>Digit(5)</th><th>Digit(6)</th><th>Digit(7)</th><th>Digit(8)</th><th>Sum</th><th>OUTPUT</th></tr>
<tr><td>4</td><td>3</td><td>1</td><td>0</td><td>2</td><td>3</td><td>1</td><td>0</td><td>30</td><td>GTIN-8</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>43102310</td></tr>
</table>
<b>One</b> mark for data entry – both sets of digits 1–7
<b>One</b> mark for both Digit(8)
<b>One</b> mark for each Sum (max <b>Two</b>)
<b>One</b> mark for both OUTPUT
</td><td>5</td></tr>
<tr><td>3(b)</td><td>
Any <b>three</b> from
1 Change first loop to 8 iterations
2 Check that the input Digit(8) is equal to the calculated Digit(8)…
3 … if equal output check digit correct
4 … otherwise output check digit incorrect

<b>Or</b>

1 Change first loop to 8 iterations
2 Put all 8 digits through the algorithm to calculate Sum …
3 … if MOD(Sum,10) is equal to zero, check digit correct
4 … otherwise output check digit incorrect
</td><td>3</td></tr>
</table>

Question	Answer	Marks
4	**One** mark for each (max **three**) *10.00* boundary/erroneous data // the price should be rejected // value is out of range *9.99* boundary/extreme/normal data // the prices should be accepted // value is within normal range *ten* erroneous/abnormal data // input should be rejected // value is wrong type	**3**

Question	Answer	Marks
5	There are many possible answers. e.g.: Totalling is used to sum a list of numbers (1) Counting is used to find how many numbers/items there are in a list. (1) Totalling example (1) e.g. `Total = Total + Number` Counting example (1) e.g. `Counter = Counter + 1`	**4**

Question	Answer	Marks
6(a)	Fields 5 Records 8	**2**
6(b)	Any **two** from: Length check Type check Presence check Format check	**2**

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>6(c)</td><td>
<table>
<tr><td>Field:</td><td>Type</td><td>Sold Out</td><td>Date</td><td>Title</td></tr>
<tr><td>Table:</td><td>PERFORMANCE</td><td>PERFORMANCE</td><td>PERFORMANCE</td><td>PERFORMANCE</td></tr>
<tr><td>Sort:</td><td></td><td></td><td></td><td></td></tr>
<tr><td>Show:</td><td>□</td><td>□</td><td>☑</td><td>☑</td></tr>
<tr><td>Criteria:</td><td>Like “Jazz”</td><td>False</td><td></td><td></td></tr>
<tr><td>or:</td><td></td><td></td><td></td><td></td></tr>
</table>
<b>One</b> mark per correct column.</td><td><b>4</b></td></tr>
</table>