Cambridge IGCSE

Cambridge Assessment International Education
Cambridge International General Certificate of Secondary Education

COMPUTER SCIENCE **0478/21**

Paper 2 **October/November 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 October/November 2018 series for most Cambridge IGCSE™, Cambridge International A and AS Level components and some Cambridge O Level components.

This syllabus is approved for use in England, Wales and Northern Ireland as a Cambridge International Level 1/Level 2 Certificate.

This document consists of **14** printed pages.

Cambridge Assessment International Education

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)	1 mark for any meaningful array name related to Task 1 × 2
	1 mark for correct data type **AND** use related to Task 1 × 2
	
	These are examples, many alternatives are valid.
	
	`ItemCode`
	…string
	…to store the item codes
	
	`MenuItem`
	…string
	…to store the menu items
	
	`Price`
	…real
	…to store the prices
		4

Question	Answer	Marks

1(a)(ii)	1 mark for any meaningful variable name related to Task 2 × 2
	1 mark for correct data type **AND** use related to Task 2 × 2, e.g.
	
	These are examples, many alternatives are valid.
	
	`ItemCode`
	…string
	…to allow items from menu to be input
	
	`ItemCost`
	…real
	…to hold price of current menu item
	
	`TotalCost`
	…real
	…to store current total price of the order
	
	`Quantity`
	…integer
	…to store the quantity of an item ordered
		4

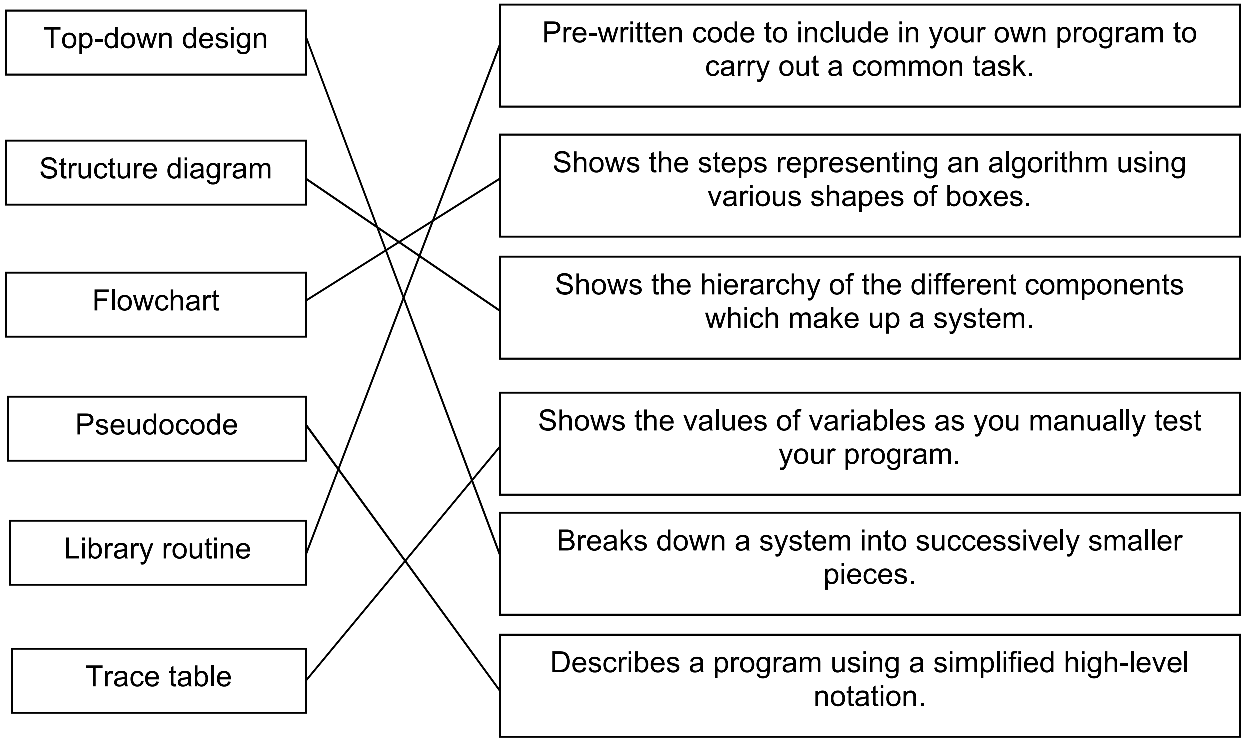
Question	Answer	Marks
1(b)	Any **five** from: • Initialisation • Loop for entry order • Input of item code and quantity • Calculation of total price • Method of termination of loop e.g. use of flag • Generation of unique order code • Display Order… • … display complete with unique order code, menu items, quantities, prices, total cost of order	**5**

Example algorithm

```
Total_Cost ← 0
Order_Complete ← FALSE
Order_Item ← 0
Order_Number ← 0
Daily_Order_Counter ← 1
// The following would be repeated throughout the day
REPEAT
  OUTPUT "Enter item code or X to finish"
  INPUT Item_Code[Order_Item]
  IF Item_Code[Order_Item] <> "X"
    THEN
      Menu_Count ← 0
      Menu_Flag ← FALSE
      REPEAT
        IF Menu_Code[Menu_Count] = Item_Code[Order_Item]
          THEN
            Menu_Place[Order_Item] ← Menu_Count
            Menu_Flag ← TRUE
            Total_Cost ← Total_Cost + Menu_Price[Menu_Count]
          ELSE
            Menu_Count ← Menu_Count + 1
        ENDIF
      UNTIL Menu_Flag ← TRUE
      OUTPUT "How many would you like? "
```

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>1(b)</td><td><pre>
      INPUT Item_Quantity[Order_Item]
      Order_Item ← Order_Item + 1
    ELSE
      Order_Complete ← TRUE
      Order_Number ← Order_Number + 1
  ENDIF
UNTIL Order_Complete = TRUE
OUTPUT "Order Number ", Order_Number
Counter ← 0
WHILE Item_Code[Counter] <> "X" DO
  Menu_Count ← Menu_Place[Counter]
  OUTPUT Item_Code[Counter], " ", Menu_Item{Menu_Count], " ",
  Price[Menu_Count], " ",Item_Quantity[Counter]
  Counter ← Counter + 1
ENDWHILE
OUTPUT "Total cost of order = ", Total_Cost
Daily_Order_Code[Daily_Order_Counter] ← Order_Number
Daily_Total_Cost_of_Order[Daily_Order_Counter] ← Total_Cost
</pre></td><td></td></tr>
<tr><td>1(c)</td><td>Any <b>four</b> from:
• Explanation of using an input prompt and statement to enter the percent of the takings that are profit
• Explanation of how the total takings are calculated
• Explanation of how the percentage profits are calculated (from the input value)
• Explanation of how the output is done // Explanation showing correct output statement with the daily takings, profit and percent value used in the calculation</td><td>4</td></tr>
</table>

Question	Answer	Marks
1(d)	1 mark for each correct test data item and related reason for **Task 3** which requires a percentage to be input (Answers **MUST** relate to pre-release task) e.g. Test data: -10 Reason: To check that negative values of percentage are rejected (or can identify a loss) Test data: 7.5 Reason: To check that normal data is accepted Test data: 7Percent Reason: To check that incorrect data / data types are rejected	**3**

Question	Answer	Marks
	Section B	
2	1 mark for each correct line, maximum 5 marks **Term** **Description**  Top-down design Structure diagram Flowchart Pseudocode Library routine Trace table Pre-written code to include in your own program to carry out a common task. Shows the steps representing an algorithm using various shapes of boxes. Shows the hierarchy of the different components which make up a system. Shows the values of variables as you manually test your program. Breaks down a system into successively smaller pieces. Describes a program using a simplified high-level notation.	**5**

Question	Answer	Marks
3	1 mark for description 1 mark for example, e.g. To test if the value input falls between a given upper bound and a given lower bound … If a month has to be input using an integer, it must be between 1 and 12 inclusive. 1 mark for description 1 mark for example, e.g. To test if the data input is over/under a certain number of characters … An international telephone number can be no longer than 15 digits. 1 mark for description 1 mark for example, e.g. To test if the input is of the correct <u>data</u> type … If the input is expecting integer(s) to be entered, it will not permit a string to be entered.	**6**

Question	Answer	Marks
4(a)	1 mark for each point: • Expects 50 numbers to be input • Totals the numbers as they are entered / carries out a running total • Outputs the result after the numbers have all been entered	**3**

4(b)	1 mark for each point (max 3 marks):
	• Correct initialisation of counter for REPEAT or WHILE loop …
	• … Correct loop statements and counter increment
	• … Correct statements inside loop
	• … Correct statements outside loop
	
	e.g.
	`Total ← 0`
	`Count ← 1`
	`WHILE Count <= 50 DO`
	`   INPUT Num`
	`   Total ← Total + Num`
	`   Count ← Count + 1`
	`ENDWHILE`
	`OUTPUT Total`
	
	Accept alternative correct ranges e.g.
	`Count ← 0`
	`WHILE Count < 50 DO`
		3

Question	Answer	Marks
4(b)	<code>Total ← 0</code> <code>Count ← 0</code> <code>REPEAT</code> <code> INPUT Num</code> <code> Total ← Total + Num</code> <code> Count ← Count + 1</code> <code>UNTIL Count = 50</code> <code>OUTPUT Total</code> Accept alternative correct ranges e.g. <code>Count ← 1</code> <code>UNTIL Count > 50</code>	
4(c)	1 mark for each correct point in description, e.g. • Use a variable for the counter upper limit… • … that is input by the user. or • loop using a condition control … • … until condition is met	2

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>5(a)</td><td>
<table>
<tr><th>Flag</th><th>TestNum</th><th>Num</th><th>OUTPUT</th></tr>
<tr><td>True</td><td>7</td><td>6</td><td></td></tr>
<tr><td></td><td></td><td>5</td><td></td></tr>
<tr><td></td><td></td><td>4</td><td></td></tr>
<tr><td></td><td></td><td>3</td><td></td></tr>
<tr><td></td><td></td><td>2</td><td></td></tr>
<tr><td></td><td></td><td>1</td><td>7</td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td colspan="3"><------------------- 1 Mark -----------------------></td><td><- 1 Mark - ></td></tr>
</table>
</td><td>2</td></tr>
</table>

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>5(b)</td><td>
<table>
<tr><th>Flag</th><th>TestNum</th><th>Num</th><th>OUTPUT</th></tr>
<tr><td>True</td><td>6</td><td>5</td><td></td></tr>
<tr><td></td><td></td><td>4</td><td></td></tr>
<tr><td></td><td></td><td>3</td><td></td></tr>
<tr><td>False</td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td colspan="4"><---------------- 1 Mark ------------->←---------1 Mark ------------------- ></td></tr>
</table>
</td><td>2</td></tr>
<tr><td>5(c)</td><td>1 mark for correct purpose e.g.
Works out if the number entered is a prime number.</td><td>[1]</td></tr>
</table>

Question	Answer	Marks
6(a)	**Field** / **Data type**: PCID – Text ScreenSize – Number Type – Text Price – Currency 2 marks for 4 correct data types 1 mark for 2 or 3 correct data types	**2**
6(b)	1 mark correct Fields included 1 mark correct Table and Show on all fields used 1 mark for correct sort, must be descending on Price 1 mark for correct criteria for the fields	**4**

6(a)

Field	Data type
PCID	Text
ScreenSize	Number
Type	Text
Price	Currency

6(b)

Field:	PCID	ScreenSize	RAM	Type	HDD(Gb)	Price
Table:	PCSTOCK	PCSTOCK	PCSTOCK	PCSTOCK	PCSTOCK	PCSTOCK
Sort:						Descending
Show:	☑	☑	☑	☑	☑	☑
Criteria:				="DT"	>1000	
or:						