Cambridge O Level

Cambridge Assessment International Education
Cambridge Ordinary Level

COMPUTER SCIENCE **2210/23**

Paper 2 **October/November 2019**

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 2019 series for most Cambridge IGCSE™, Cambridge International A and AS Level components and some Cambridge O Level components.

© UCLES 2019

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 2019

Question	Answer	Marks
1(a)	Two examples of: Any meaningful name for an array related to **Task 1** – **one** mark, e.g. • `TileDescription` • `TilePrice` • `TileCode` Correct data type **and** purpose related to **Task 1** – **one** mark, e.g. • … string to store the name / description of the tiles • … real to store the price of the tiles • … string to store the tile codes	**4**
1(b)	Any meaningful name for a variable related to **Task 2** – **one** mark, e.g. • `NumberOfWalls` • `TotalArea` • `TotalCost` Relevant data type for the variable related to **Task 2** – **one** mark, e.g. • … Integer • … Real Relevant purpose for the variable related to **Task 2** – **one** mark, e.g. • … to store the number of walls that need to be tiled • … to store the total area / cost **One** mark for a correct reason, e.g. • Variables allow the storage of values within a program that may change as the program runs // Variables are used to store values that are input or calculated	**4**

Question	Answer	Marks

1(c)	**Six** from:
	MP1 Input for height and width of the wall to tile and tile code
	MP2 Prompts for all inputs seen
	MP3 Validation of height and width
	MP4 Reasonable attempt at validation of tile code
	MP5 Calculation of the area of the wall
	MP6 Calculation of the number of boxes rounded up
	MP7 Looking up the cost of the tiles
	MP8 Calculation of the cost of the boxes of the tiles
	MP9 Output of area of the wall, the number of boxes and cost of tiles needed
	MP10 Appropriate message(s) with output
	
	Example algorithm
	`REPEAT`
	`  OUTPUT "Please enter height of wall "`
	`  INPUT Height`
	`UNTIL Height > 0`
	
	`REPEAT`
	`  OUTPUT "Please enter width of wall "`
	`  INPUT Width`
	`UNTIL Width > 0`
	
	`REPEAT`
	`  Found ← FALSE`
	`  Counter ← 0`
	`  REPEAT`
	`  OUTPUT "Please enter tile identification code "`
	`  INPUT MyTileId`
	`  IF MyTileId = TileCode(Counter)`
	`  THEN`
	`  MyCost ← TilePrice(Counter)`
	`  Found ← TRUE`
	`  ENDIF`
	`  Counter ← Counter + 1`
	`  UNTIL Found OR Counter = 10`
	`UNTIL Found`
	
	`Area ← Height * Width`
	`NoBoxes ← Int(Round (Area + 0.5))`
	`// Rounds up to nearest Integer`
	`TotalPrice ← NoBoxes * MyCost`
	
	`OUTPUT ("Area of wall is ", Area, " metres")`
	`OUTPUT ("Number of boxes of tiles is ", Area, " NoBoxes)`
	`OUTPUT ("Cost of Tiles is ", TotalPrice, " dollars"`
		6

© UCLES 2019

Question	Answer	Marks
1(d)	**Three** from: • Adding an extra user input for number of walls … • … using this value as a loop counter • … so that separate inputs of height and width can be made for each wall • Calculating a running total for the final area using the results of the area calculation for each wall • Rounding the final area or using previously rounded areas for the final total • Displaying with a suitable message final area, final number of boxes of tiles required and final cost If only program statements given with no explanation, **zero** marks.	**3**
1(e)	**Three** from: • Check that only numbers are accepted // type check • Check that the values are within boundaries // range check • Check that a value has been entered // presence check • Using an IF / conditional statement • Identification of suitable lower value / acceptable value • Identification of suitable upper value / unacceptable value • Alerting the user with an error message if the input is unacceptable	**3**

© UCLES 2019

Question	Answer	Marks
2	**Two** from • Sub-program / system not the whole program / system • To perform a frequently used operation within a program • That can be called when needed • That can be reused by another program	**2**

Question	Answer	Marks
3	• `FOR (… TO … NEXT)` loop • `WHILE (… DO … ENDWHILE)` loop • `REPEAT (… UNTIL)` loop	**3**

Question	Answer	Marks
4(a)	• Conditional / selection statement	**1**

4(b)	**Four** from:
	MP1 CASE statement with identifier Response
	MP2 Correct structure used for choices…
	MP3 …. correct statements used for choices
	MP4 OTHERWISE and correct statement
	MP5 Single ENDCASE included
	e.g.
	`CASE OF Response // CASE Response OF`
	`  1 : X ← X + Y`
	`  2 : X ← X - Y`
	`  3 : X ← X * Y`
	`  4 : X ← X / Y`
	`  OTHERWISE OUTPUT "No response"`
	`ENDCASE`
		4

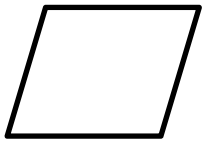
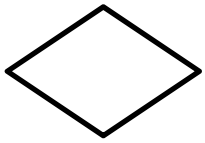
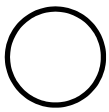
Question	Answer	Marks
5(a)		**6**

First	Last	UserIn	Middle	Found	OUTPUT
0	16			FALSE	
0	16	10	8	FALSE	
0	7	10	3	FALSE	
4	7	10	5	TRUE	TRUE

One mark per correct column

Question	Answer	Marks
5(b)	**Two** from: • Search for the value input … • … using an array… • … of sorted data	**2**

Question	Answer	Marks
6	**One** mark for each correct symbol **and** name / description / example of use (maximum four marks)	**4**

Symbol	Description of use
	Terminator – start / end the flowchart
	Process – to show calculations, etc.
	Input / Output
	Decision – to show condition
	Continuation – to extend the flowchart and allow it to join up

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>7(a)</td><td>• Number of Fields: 4
• Number of Records: 32</td><td>2</td></tr>
<tr><td>7(b)</td><td>• No field is suitable as a primary key …
• … because none of the data would be unique // duplicates could occur</td><td>2</td></tr>
<tr><td>7(c)</td><td>
<table>
<tr><td>Field:</td><td>FirstName</td><td>LastName</td><td>Y10TestScore</td><td></td></tr>
<tr><td>Table:</td><td>MARKBOOK</td><td>MARKBOOK</td><td>MARKBOOK</td><td></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>>=50</td><td></td></tr>
<tr><td>or:</td><td></td><td></td><td></td><td></td></tr>
</table>
One mark for each completely correct column down to and including ‘Show’ row (maximum three marks)
One mark for correct search criteria rows
</td><td>4</td></tr>
</table>