Cambridge O Level

Cambridge Assessment International Education
Cambridge Ordinary Level

COMPUTER SCIENCE	**2210/22**
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 document consists of **11** 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.

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td></td><td><b>Section A</b></td><td></td></tr>
<tr><td>1(a)(i)</td><td>Many correct answers, they must be meaningful. The name is an example only.
1 mark per bullet point
<table>
<tr><td>Variable name</td><td><code>Winner4to6</code></td></tr>
<tr><td>Data type</td><td>string</td></tr>
<tr><td>Use</td><td>storing the unique id of the runner age 4 to 6 with the fastest time</td></tr>
</table></td><td><b>3</b></td></tr>
<tr><td>1(a)(ii)</td><td>Many correct answers, they must be meaningful. The names are examples only.
1 mark per bullet point
5 arrays seen
At least one suitable name e.g. <code>IdNumber</code>
At least one suitable datatype e.g. string
At least one suitable use e.g. to store the unique identification number
All 5 names, datatypes and uses correct
Array examples
<table>
<tr><th>Array name</th><th>datatype</th><th>Use</th></tr>
<tr><td><code>IdNumber</code></td><td>integer</td><td>unique identification number</td></tr>
<tr><td><code>Name</code></td><td>string</td><td>name of child</td></tr>
<tr><td><code>Age</code></td><td>integer</td><td>age of child in years</td></tr>
<tr><td><code>NumRuns</code></td><td>integer</td><td>number of 2 kilometre runs completed</td></tr>
<tr><td><code>PB</code></td><td>real</td><td>fastest time run</td></tr>
</table></td><td><b>5</b></td></tr>
</table>

Question	Answer	Marks
1(b)	Many correct answers. 1 mark for each correct point (max 4). Ensuring uniqueness e.g. Search list of previously stored identification numbers (1) To check it has not already been used. (1) Calculation and addition of check digit Attempt at calculation of check digit from the first three digits // calculation from all four digits to obtain zero remainder (1) Adding the calculated check digit as a fourth digit // confirming the check digit is valid. (1)	**4**
1(c)	Any **four** from: 1 Initialisation//Setting up array for race times for this race 2 Loop for all runners 3 Input ID 4 Check for valid ID number 5 Input start time and finish time 6 Calculate run time 7 Store run time in suitable array//Use ID index to store run time in appropriate position 8 Store ID in suitable array//Find index for runner	**4**

Question	Answer	Marks
1(c)	Sample answer:	

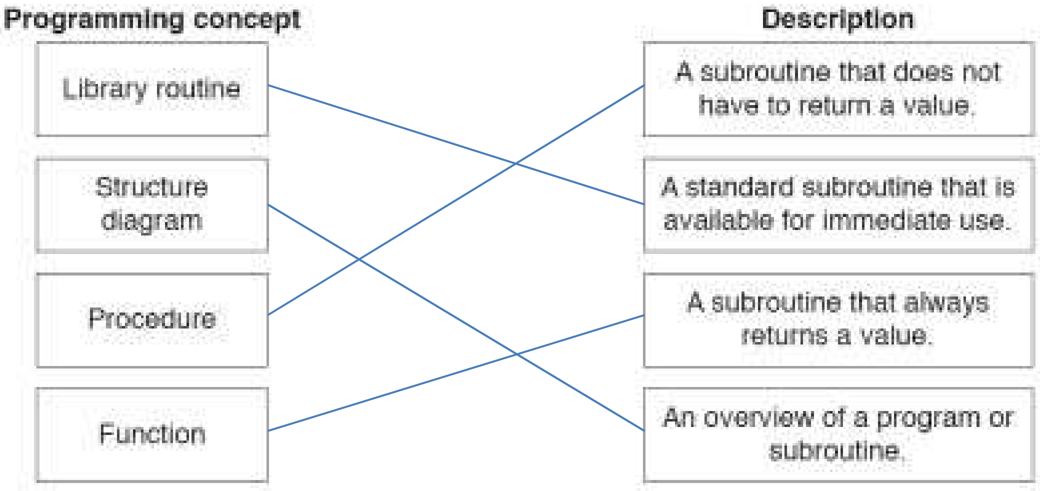
```
Counter ← 1
Found ← FALSE
WHILE NOT Found = TRUE
   OUTPUT "Please enter ID number use 9999 to finish "
   INPUT ParkrunID
   Search ← 1
   REPEAT
      IF ParkrunID = IdNumber[search] THEN Found ← TRUE
      Search ← Search + 1
   UNTIL Search = 21 OR Found = TRUE
ENDWHILE
RunnerID[Counter] ← ParkRunID
WHILE ParkRunID <> 9999
   INPUT StartTime
   INPUT FinishTime
   RunnerTime[Counter] ← FinishTime – StartTime
   Counter ← Counter + 1
   Found ← FALSE
   WHILE NOT Found = TRUE
      OUTPUT "Please enter ID number (use 9999 to finish) "
      INPUT ParkrunID
      Search ← 1
      REPEAT
      IF ParkrunID = IdNumber[Search] THEN Found ← TRUE
         Search ← Search + 1
      UNTIL Search = 21 OR Found = TRUE
   ENDWHILE
   RunnerID[Counter] ← ParkRunID
ENDWHILE
```

Question	Answer	Marks
1(d)	1 mark for each correct point (max 4). Explanation: 1 Search the results for this park run 2 Separate check for each age range 3 If time recorded is faster 4 … store new fastest time 5 … store the ID/Name for the runner 6 Use ID number/Index to look up name after all runners have been searched 7 Output name and fastest time Programming statements may be used but must be fully explained.	**4**

Question	Answer	Marks
	Section B	

2(a)	Any **five** from:
	1 Use of correct variables
	2 Input 3 numbers
	3 Check all 3 input numbers are different
	4 Attempt to find the largest two numbers input
	5 Correctly finding the largest two numbers
	6 Multiply their two largest numbers together and assign to variable
	7 Output the result of the multiplication
	
	Sample answer:
	`REPEAT`
	`   OUTPUT "Enter three different numbers"`
	`   INPUT Number1, Number2, Number3`
	`UNTIL Number1 <> Number2 AND Number2 <> Number3 AND Number3 <> Number1`
	`IF Number3 < Number2 AND Number3 < Number1`
	`   THEN Answer ← Number1 * Number2`
	`ENDIF`
	`IF Number2 < Number3 AND Number2 < Number1`
	`   THEN Answer ← Number1 * Number3`
	`ENDIF`
	`IF Number1 < Number2 AND Number1 < Number3`
	`   THEN Answer ← Number2 * Number3`
	`ENDIF`
	`OUTPUT "Answer = ", Answer`
		5

2(b)	There are many correct answers. E.g.: 7, 7, 7 (1 mark) … should be rejected as numbers are equal (1 mark) 7, 8, 9 (1 mark) … normal data answer should be 72 (1 mark)	4

Question	Answer	Marks
3	1 mark for each correct line (max 3) Each box must have only one connection.  Programming concept Library routine Structure diagram Procedure Function Description A subroutine that does not have to return a value. A standard subroutine that is available for immediate use. A subroutine that always returns a value. An overview of a program or subroutine.	**3**

Question	Answer	Marks
4	Answers must be given in context. There are many possible answers. E.g.: Selection use of `IF` statement to check the values of the meter readings (1 mark) `IF Reading > 400 and Reading < 900 THEN …` (1 mark) Repetition use of `FOR` loop to check all 2000 meter readings (1 mark) `FOR Meter = 1 TO 2000 … NEXT` (1 mark)	**4**

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>5(a)</td><td>
<table>
<tr><th>Height</th><th>Depth</th><th>Chlorine</th><th>OK</th><th>OUTPUT</th></tr>
<tr><td>6</td><td>2.5</td><td>2</td><td>True</td><td>Pool OK to use</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
</table>
<table>
<tr><th>Height</th><th>Depth</th><th>Chlorine</th><th>OK</th><th>OUTPUT</th></tr>
<tr><td>4</td><td>3</td><td>1.5</td><td>True</td><td>Water too deep</td></tr>
<tr><td></td><td></td><td></td><td>False</td><td></td></tr>
</table>
<table>
<tr><th>Height</th><th>Depth</th><th>Chlorine</th><th>OK</th><th>OUTPUT</th></tr>
<tr><td>6</td><td>3.5</td><td>4</td><td>True</td><td>Water too deep</td></tr>
<tr><td></td><td></td><td></td><td>False</td><td>Too much chlorine add more water</td></tr>
</table>
1 mark for first 4 columns in each trace table (max 3)
1 mark for the output in each trace table (max 3)
</td><td><b>6</b></td></tr>
<tr><td>5(b)</td><td>Any <b>one</b> from:
Cannot add more water if the water is too deep
No validation e.g. allows a negative height/depth/amount of chlorine
Tells you to add chlorine when there is no water
Runs only once</td><td><b>1</b></td></tr>
</table>

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>6(a)</td><td>Many correct answers, an example is given. 1 mark for each correct row (max 4).
<table><tr><th>Field</th><th>Data type</th></tr><tr><td>Reference Number</td><td>Text</td></tr><tr><td>Size</td><td>Text</td></tr><tr><td>Type</td><td>Text/Boolean</td></tr><tr><td>Price in $</td><td>Number/Currency</td></tr></table></td><td>4</td></tr>
<tr><td>6(b)</td><td>1 mark per bullet:
Incorrect field name for Reference Number
Incorrect criteria for Price in $ should be <
Type not checked</td><td>3</td></tr>
</table>