Cambridge Assessment International Education

Cambridge IGCSE™

COMPUTER SCIENCE **0478/22**

Paper 2 Problem-solving and Programming **October/November 2022**

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 2022 series for most Cambridge IGCSE™, Cambridge International A and AS Level components and some Cambridge O Level components.

[Turn over

PUBLISHED

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.

Please note the following further points:

The words in **bold** in the mark scheme are important text that needs to be present, or some notion of it needs to be present. It does not have to be the exact word, but something close to the meaning.

If a word is underlined, this **exact** word must be present.

A single forward slash means this is an alternative word. A double forward slash means that this is an alternative mark point.

Ellipsis (…) on the end of one-mark point and the start of the next means that the candidate **cannot** get the second mark point without being awarded the first one. If a mark point has an ellipsis at the beginning, but there is no ellipsis on the mark point before it, then this is just a follow-on sentence and **can** be awarded **without** the previous mark point.

Question	Answer	Marks
	Section A	
1(a)(i)	Many correct answers, the name used must be meaningful. The name given is an example only. **One** mark per mark point, max **two** • Constant `NumberDays` … • Value … 14 **Task 1** – setting up the booking system Set up suitable data structures to store the car licence numbers and names of visitors who have booked car parking spaces. The data structures should have sufficient capacity to store data for each of the 20 parking spaces for a static period of two weeks. Allow a visitor to request a parking space on any day within the two-week period by entering a number between 1 and 14, inclusive. The system will check that there are spaces available on the day requested, and if so, will ask the visitor to enter their name and car licence number. This data will be stored in the data structures representing the first available parking space for the day requested. The visitor will be told the number of their parking space. At the end of the two-week period, allow all of the data to be deleted ready for the next two-week period.	**2**
1(a)(ii)	Many correct answers, the name used must be meaningful. The name given is an example only. **One** mark per mark point, max **two** • Array `LicenceNumbers` … • Use … storing the licence numbers of the cars to be parked	**2**

Question	Answer	Marks

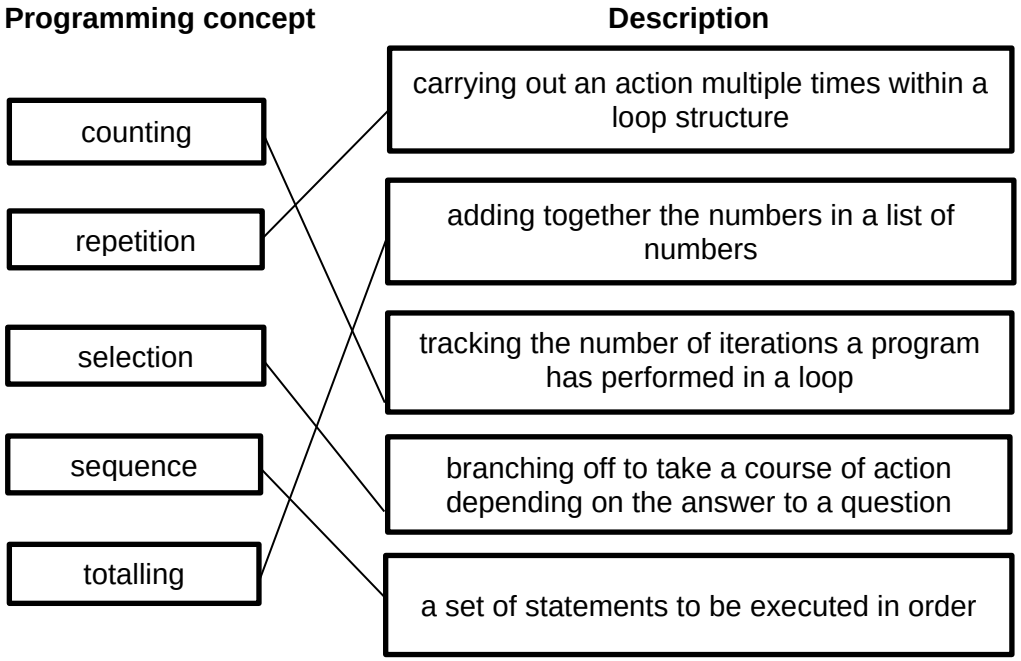
1(b)	**One** mark per mark point, max **two**
	
	MP1 use a (range) check to check for values between 1 and 14
	MP2 use a (type) check to ensure an integer is entered
	MP3 … using an (appropriate) `IF`/conditional statement
	MP4 outputs an error message if a new input is required
	MP5 re-enter data using `WHILE`/`REPEAT` loop until a valid entry has been made
	
	One mark
	MP6 appropriate line of code / construct to answer the question
	
	Example code
	`INPUT Day`
	`WHILE Day < 1 OR Day > 14 DO`
	`  OUTPUT "You must enter a number between 1 and 14 inclusive"`
	`  INPUT Day`
	`ENDWHILE`
		3

Question	Answer	Marks
1(c)	**One** mark per mark point, max **six** MP1 input for day number or to choose accessible parking … MP2 … **both** inputs with appropriate prompts MP3 conditional statement to check if accessible parking is required MP4 initialisation of variable for the first accessible space MP5 checking if current array element is an available space MP6 correct loop to check all array elements for the day MP7 … until an available space is found (or not) MP8 input car licence number and name (with or without prompts) MP9 … and store them in the correct array position(s) MP10 appropriate output for space available or no space available MP11 appropriate output for allocated parking space details **Task 2** – adding accessible parking spaces The visitor car park booking system is to be re-designed to offer accessible parking. Spaces 1 to 5 are named accessible spaces. Spaces 6 to 20 are named general spaces. Extend your program in **Task 1** so that: • when a visitor requests a parking space, they are additionally asked if they need an accessible space – if so, they are allocated the first available space beginning at space 1 and finishing at space 20 – if **not**, they are allocated the first available space beginning at space 20 and finishing at space 6. The system must work so that visitors requiring accessible parking may be allocated any of the 20 spaces, but visitors who do **not** need accessible parking may only be allocated general spaces.	**6**

Question	Answer	Marks

1(c)	**Example answer**
	
	`//Assume LicenceNumber array has been initialised with the null string`
	`OUTPUT "Enter the number of the day in which you require parking "`
	`INPUT Day`
	`OUTPUT "Do you require an accessible space? (Y or N)"`
	`INPUT Accessible`
	`IF Accessible = "Y"`
	`  THEN`
	`    SpaceAllocated ← FALSE`
	`    Space ← (Day - 1) * 20`
	`    REPEAT`
	`        IF LicenceNumbers[Space] = ""`
	`          THEN`
	`            OUTPUT "Enter your car licence number "`
	`            INPUT LicenceNumbers[Space]`
	`            SpaceAllocated ← TRUE`
	`        ENDIF`
	`        Space ← Space + 1`
	`    UNTIL SpaceAllocated OR Space = (Day - 1) * 20 + 20`
	`    IF NOT SpaceAllocated`
	`      THEN`
	`        OUTPUT "No space available"`
	`      ELSE`
	`        OUTPUT "You are allocated space: ", Space – (Day – 1) * 20`
	`    ENDIF`
	`ENDIF`	

Question	Answer	Marks
1(d)	Explanation of **how** the following could be done: **One** mark per mark point, max **three** MP1 change the max constant to 560 / 28 MP2 change the output message for the user to say that the range of days entered must be from 1 to 28 MP3 alter the validation to allow an input of 1 to 28 MP4 change the size of the array that stores licence numbers / names so that it can hold the additional data MP5 ensure that all loops that access the stored data for retrieval or erasing of data include the new full range	**3**
1(e)	Explanation of **how** the following could be done: **One** mark per mark point, max **four** MP1 initialise counting variable(s) … MP2 … for accessible spaces **and** general spaces before any spaces have been allocated MP3 increment the counter(s) … MP4 … for accessible spaces **and** general spaces MP5 maintain **both** counters for the full two-week cycle MP6 output to show the total number of accessible and general spaces allocated so far, with suitable messages • **Task 3** – working out car park usage statistics • Extend the program to enable the following statistics to be counted and output on request: • The number of accessible spaces used on any of the 14 days. • The number of general spaces used on any of the 14 days. • The total number of spaces used on any of the 14 days. • The number of accessible spaces used in the whole 14-day period. • The number of general spaces used in the whole 14-day period. • The total number of spaces used in the whole 14-day period.	**4**

Question	Answer	Marks
	Section B	
2	**One** mark for each correct line, max **four** 	**4**

Question	Answer	Marks
3	**One** mark per mark point, max **three** MP1 verification is used to make sure the items in stock do not change from the original when they are input // verification is used to make sure the items in stock do not change from what was intended to be input // verification is used to make sure the items are **accurately copied** MP2 enter each item in stock twice / double entry // visual check MP3 matching description of the type of check stated in MP2 **Example answers** Double entry [1] enter data twice **and** only accept identical values [1] Visual check [1] look at the data that has been entered **and** confirm it matches [1]	**3**

Question	Answer	Marks
4	**One** mark per mark point, max **two** • type of test data … • … description of test data **Example answers** Normal data (1) data that would be accepted by the program (1) Boundary / extreme data (1) data that is **on** the acceptable limits (1)	**2**

Question	Answer	Marks

5(a)	**One** mark per mark point, max **four**
	
	• Line 09 / `Higher[HighList] ← MarksEntry`
	should be `Higher[HighList] ← Mark`
	
	• Line 15 / `MidList ← MidList`
	should be `MidList ← MidList + 1`
	
	• Line 17 / `Lower[HighList] ← Mark`
	should be `Lower[LowList] ← Mark`
	
	• Line 22 / `NEXT MarksEntry = 500`
	should be `UNTIL MarksEntry = 500`
	
	Corrected algorithm
	`01 HighList ← 0`
	`02 MidList ← 0`
	`03 LowList ← 0`
	`04 MarksEntry ← 0`
	`05 REPEAT`
	`06  INPUT Mark`
	`07  IF Mark >= 80`
	`08  THEN`
	`09  Higher[HighList] ← Mark`
	`10  HighList ← HighList + 1`
	`11  ELSE`
	`12  IF Mark >= 50`
	`13  THEN`
	`14  Middle[MidList] ← Mark`
	`15  MidList ← MidList + 1`
		4

Question	Answer	Marks
5(a)	16 ELSE 17 Lower[LowList] ← Mark 18 LowList ← LowList + 1 19 ENDIF 20 ENDIF 21 MarksEntry ← MarksEntry + 1 22 UNTIL MarksEntry = 500 23 OUTPUT "You entered ", HighList, " higher marks" 24 OUTPUT "You entered ", MidList, " middle marks" 25 OUTPUT "You entered ", LowList, " lower marks"	

Question	Answer	Marks

5(b)	**One** mark per mark point, max **four**
	
	MP1 Set up a condition to end the input
	MP2 The correct placement of the condition
	MP3 Set up the test
	MP4 The correct placement of the test
	MP5 Removal of `MarksEntry` counter from the original algorithm
	
	Example answers
	
	Testing at the end of the algorithm
	`OUTPUT "Do you want to enter another mark?"`
	`INPUT AnotherMark`
	`UNTIL AnotherMark = "No"`
	should replace line 22 at end of loop
	The `MarksEntry` counter can be removed // Lines 4 and 21 are not required / can be removed
	
	Testing at the beginning of the algorithm
	`AnotherMark = "Yes"`
	`WHILE AnotherMark = "Yes" DO`
	should replace line 05 at the start of the loop
	`OUTPUT "Do you want to enter another mark?"`
	`INPUT AnotherMark`
	`ENDWHILE`
	should replace line 22 at end of loop
	The `MarksEntry` counter can be removed // Lines 4 and 21 are not required / can be removed
	
	Terminal condition
	`OUTPUT "Enter -1 to end the program"`
	should be placed before the loop and / or before the input in 06
	`IF MARK <> -1 THEN`
	should be placed between lines 06 and 07
	The `MarksEntry` counter can be removed // Lines 4 and 21 are not required / can be removed
	`UNTIL Mark = -1` should be placed at line 22
		4

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

One mark per mark point, max **five**

MP1 correct `Counter` and `Number` columns
MP2 correct `Hundreds` column
MP3 correct `Temp` and `Tens` columns
MP4 correct `Units` column
MP5 correct OUTPUT column

<table>
<tr><th>Counter</th><th>Number</th><th>Hundreds</th><th>Temp</th><th>Tens</th><th>Units</th><th>OUTPUT</th></tr>
<tr><td>0</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>1</td><td>97</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>2</td><td>876</td><td>8</td><td>76</td><td>7</td><td>6</td><td>Hundreds: 8 Tens: 7 Units: 6</td></tr>
<tr><td>3</td><td>4320</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>4</td><td>606</td><td>6</td><td>6</td><td>0</td><td>6</td><td>Hundreds: 6 Tens: 0 Units: 6</td></tr>
<tr><td>5</td><td>9875</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>6</td><td>42</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>7</td><td>124</td><td>1</td><td>24</td><td>2</td><td>4</td><td>Hundreds: 1 Tens: 2 Units: 4</td></tr>
<tr><td>8</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></tr>
</table>

</td><td>**5**</td></tr>
</table>

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>7(a)(i)</td><td>SubjectCode Text
Exams Number

Data types must match those given in the question.</td><td>1</td></tr>
<tr><td>7(a)(ii)</td><td>The Boolean data type can only have one of two values // the Candidates field has more than two possible values.</td><td>1</td></tr>
<tr><td>7(b)</td><td>One mark per mark point, max three
• correct data
• correct layout
• correct order

Expected answer
Geography 200
Geology 80
History 250
Mathematics 350</td><td>3</td></tr>
<tr><td>7(c)</td><td>One mark per mark point, max three
• correct fieldnames and table names
• correct sort and show rows
• correct search criteria

<table>
<tr><td>Field:</td><td>SubjectCode</td><td>SubjectName</td><td>Candidates</td><td></td><td></td></tr>
<tr><td>Table:</td><td>ASSESS</td><td>ASSESS</td><td>ASSESS</td><td></td><td></td></tr>
<tr><td>Sort:</td><td></td><td></td><td>Descending</td><td></td><td></td></tr>
<tr><td>Show:</td><td>☑</td><td>☑</td><td>☑</td><td>☐</td><td>☐</td></tr>
<tr><td>Criteria:</td><td></td><td></td><td><150</td><td></td><td></td></tr>
<tr><td>or:</td><td></td><td></td><td></td><td></td><td></td></tr>
</table>
</td><td>3</td></tr>
</table>