Cambridge Assessment International Education

Cambridge IGCSE™

COMPUTER SCIENCE **0478/23**

Paper 2 Problem Solving and Programming **May/June 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 May/June 2022 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 **13** printed pages.

© UCLES 2022

[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.

© UCLES 2022

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 2022

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.

© UCLES 2022

Question	Answer	Marks
	Section A	
1(a)	Many correct answers, the names used must be meaningful. The names given are examples only. **One** mark per mark point, max **four** • Variable `ClientName` • Use Storing the name of the person making the booking • Array `BookingsLarge[ ]` • Use Storing the bookings for the large meeting room **Task 1** – setting up a booking system structure for the meeting rooms Set up suitable data structures for each of the three meeting rooms to store: • when it is booked during a fixed eight-week period • the client's name (the person making the booking) • a unique booking code • the cost of the booking. **Task 2** – booking a meeting room Extend the program in **Task 1** to enable bookings to be made so that the client enters their name, the meeting room required and the day of the booking. After the data has been entered, the program should check if the requested day is available for the required meeting room and if **not**, the client should be allowed to enter an alternative day or exit the program. If the requested day is available, the booking details and cost of the booking should be output for the client to confirm. Once confirmed, a unique booking code should be generated and stored in both the appropriate meeting room data structure and the unique booking code data structure. The client's name and cost of the booking should be stored in the appropriate data structures set up in **Task 1**. Bookings of more than one day must be entered as separate single day bookings.	**4**

© UCLES 2022

Question	Answer	Marks
1(b)	**One** mark per mark point, max **two** MP1 input a piece of normal test data that should be accepted // use Large, Small1 or Small2 to check that these data are accepted // use a menu and check the input matches the available options MP2 Input a piece of erroneous test data that should be rejected // entry of anything that is not Large, Small1 or Small2 should be rejected // anything not on the menu should be rejected Explanation Task 2	**2**
1(c)	**One** mark per mark point, max **four** MP1 introduce a (new) variable/array for the number of days for the booking // introduce a (new) variable/array for the number of days for booking the same room // identify the number of days the same room has been booked by the same client MP2 use a conditional/IF statement to check if the length of the booking is 2 to 6 inclusive / 2, 3, 4, 5, or 6 //Use a CASE statement MP3 … if it is, get the daily rate for the room booking MP4 … multiply the number of days for the booking by the room rate MP5 … multiply the total cost of the booking by 70%/.7// reduce the total cost by 30% Explanation Task 2	**4**

© UCLES 2022

Question	Answer	Marks

1(d)	**One** mark per mark point, max **six**
	
	MP1 output all relevant input data using suitable variables for client name, meeting room choice, the booking day and the booking cost
	MP2 attempt to provide appropriate messages to accompany given output
	MP3 input with messages to confirm booking
	MP4 conditional statement or `WHILE` to check for a positive confirmation input
	MP5 attempt at generation of booking code
	MP6 fully unique booking code generated
	MP7 identification of meeting room using `CASE` or `IF` statements
	MP8 storage of booking code in both meeting room and booking code arrays/lists
	MP9 storage of client name and booking cost in appropriate arrays/lists
	
	Relevant parts of Task 2 for this question:
	• output the booking details and cost of the booking
	• take confirmation from client
	• generate a unique booking code
	• store the booking details.
	
	Assume the booking is for a single day and the requested booking day is available.
	
	Example answer
	`//Assume variables representing the current index position for the client //related arrays, ClientIndex, and index for date related arrays, //DateIndex, have been used`
	`OUTPUT "Your name is ", ClientName, Meeting Room ", MeetingRoom,`
	`" starting on ", StartDay, " cost of booking ", BookingCost`
	`OUTPUT "Is this correct (Y or N)"`
	`INPUT Confirm`
	`IF Confirm = "Y"`
	`THEN`
	`BookingCode ← BookingCode + 1`
	`CASE OF MeetingRoom`
	`'Lg': MeetingLarge[DayIndex] ← BookingCode`
	`'S1': MeetingSmall1[DayIndex] ← BookingCode`
	`'S2': MeetingSmall2[DayIndex] ← BookingCode`
		6

Question	Answer	Marks

1(d)	`    ENDCASE`
	`    BookingCode[ClientIndex] ← BookingCode`
	`    ClientName[ClientIndex] ← ClientName`
	`    BookingCost[ClientIndex] ← BookingCost`
	`    //No ELSE clause is specified in the task or the question`
	`    //so, ignore it if one is given`
	`ENDIF`	

1(e)	**One** mark per mark point, max **four** MP1 select the room whose records are to be searched // input the room whose records are to be searched MP2 use a suitable loop (to search through the array that stores the bookings data for that room) MP3 use an IF statement/conditional statement to check if the current array element has a booking MP4 … if not, use the array index to output the day represented MP5 check each element in the array until all array elements have been checked **Task 3** – using the booking data Extend the program in **Task 1** and **Task 2** to make use of the data that is available, to: • select a meeting room and output the days when it is free • total and output the amount of money currently taken for all three meeting rooms • check the bookings for a specific client. Only the first bullet point required for this question.	**4**

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td colspan="3">Section B</td></tr>
<tr><td>2</td><td>One mark per row, max four
<table>
<tr><th rowspan="2">Description</th><th colspan="4">Types of test data</th></tr>
<tr><th>Boundary</th><th>Erroneous / Abnormal</th><th>Extreme</th><th>Normal</th></tr>
<tr><td>test data that is always on the limit of acceptability</td><td></td><td></td><td>✓</td><td></td></tr>
<tr><td>test data that is either on the limit of acceptability or test data that is just outside the limit of acceptability</td><td>✓</td><td></td><td></td><td></td></tr>
<tr><td>test data that will always be rejected</td><td></td><td>✓</td><td></td><td></td></tr>
<tr><td>test data that is within the limits of acceptability</td><td></td><td></td><td>✓</td><td>✓</td></tr>
</table></td><td>4</td></tr>
</table>

Question	Answer	Marks
3	**One** mark per mark point, max **four** • variables are used to represent values that can change during the execution of a program // variables can be used to store the results of calculations / counting / totalling // can store values entered by the user • variable example – any data that is input into a program such as a date • constants represent values that **must stay the same throughout the execution of a program** • constant example – any value that does not change, such as Pi in mathematical formulae	**4**

© UCLES 2022

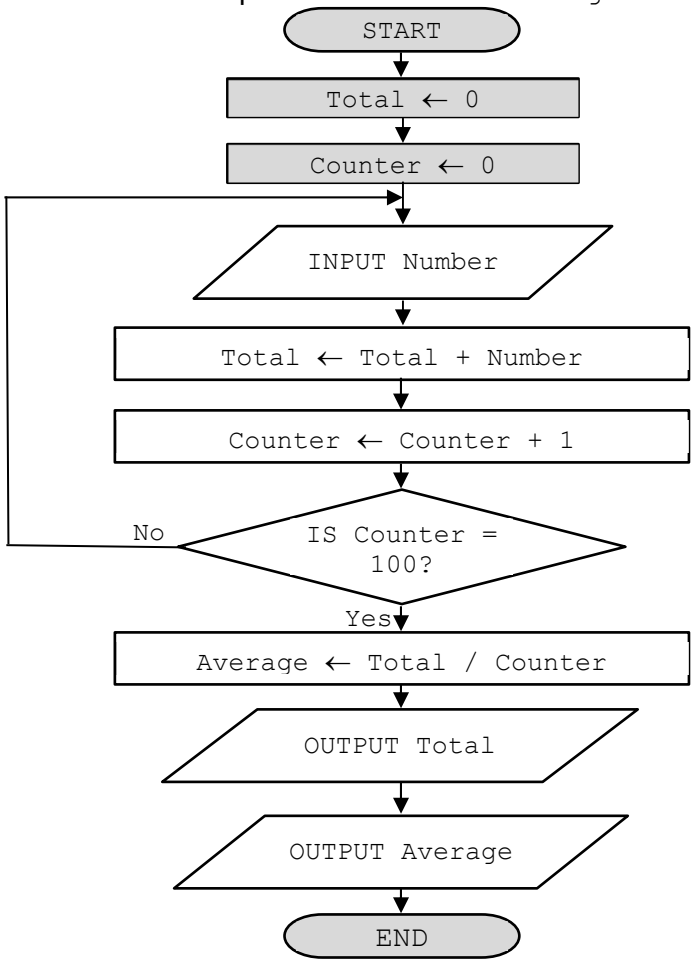
Question	Answer	Marks
4(a)	**One** mark per mark point, max **seven** MP1 correct `In` column MP2 correct `Logic` column MP3 correct `Test` column MP4 correct `Number` column MP5 correct `Store[Count]` column MP6 correct `Count` and `Limit` columns MP7 correct `Out` and OUTPUT columns	**7**

In	Logic	Test	Number	Store [Count]	Count	Limit	Out	OUTPUT
					0	5		
1	TRUE	2	9					
		3						
	FALSE							
2	TRUE	2	5					
		3		5	1			
3	TRUE	2	8					
	FALSE							
4	TRUE	2	10					
	FALSE							
5	TRUE	2	7					
		3		7	2		0	5
							1	7

© UCLES 2022

Question	Answer	Marks
4(b)	**One** mark per mark point, max **two** • to find / output prime numbers • … store prime numbers in an array	**2**
4(c)	**One** mark per mark point, max **three** MP1 insert a `WHILE` loop … // pre-condition loop MP2 … after `Input Number` MP3 … with a condition to enter the loop `Number < 3` MP4 an error message included within the loop to ask for a re-entry of `Number` MP5 …with another input prompt for `Number` MP6 `ENDWHILE` closes the loop and the program carries on from `REPEAT` in the original algorithm **OR** **One** mark per mark point, max **three** MP1 insert a `REPEAT` loop … // post-condition loop MP2 … before `Input Number` MP3 a conditional statement should be placed after `Input Number` MP4 …to check if `Number < 3` MP5 if the number entered is <3, an error message included within the loop to ask for a re-entry of `Number` MP6 `UNTIL Number >= 3` closes the loop and the program carries on from `REPEAT` in the original algorithm	**3**

© UCLES 2022

Question	Answer	Marks
5	**One** mark per mark point, max **six** MP1 input box MP2 correct totalling using `Total` MP3 correct counting using `Counter` MP4 correct conditional statement for `Counter` MP5 correct calculation of `Average` MP6 correct outputs of `Total` and `Average` 	**6**

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>6(a)</td><td>Larger</td><td>1</td></tr>
<tr><td>6(b)</td><td>One mark per mark point, max three

• correct fields correctly named and table names
• correct sort and show box rows
• correct search criteria

<table>
<tr><td>Field:</td><td>PlanetName</td><td>Larger</td><td>YearLength</td><td></td><td></td></tr>
<tr><td>Table:</td><td>PLANETS</td><td>PLANETS</td><td>PLANETS</td><td></td><td></td></tr>
<tr><td>Sort:</td><td>Ascending</td><td></td><td></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>Yes</td><td>>365</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>