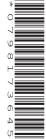# Cambridge IGCSE™

CANDIDATE NAME

CENTRE NUMBER

CANDIDATE NUMBER

COMPUTER SCIENCE **0478/22**

Paper 2 Problem-solving and Programming **February/March 2021**

1 hour 45 minutes

You must answer on the question paper.

No additional materials are needed.

INSTRUCTIONS

- Answer **all** questions.
- **Do not attempt Tasks 1, 2 and 3** in the copy of the pre-release material on page 2; these are for information only.
- Use a black or dark blue pen. You may use an HB pencil for any diagrams or graphs.
- Write your name, centre number and candidate number in the boxes at the top of the page.
- Write your answer to each question in the space provided.
- Do **not** use an erasable pen or correction fluid.
- Do **not** write on any bar codes.
- Calculators must **not** be used in this paper.

INFORMATION

- The total mark for this paper is 50.
- The number of marks for each question or part question is shown in brackets [ ].
- No marks will be awarded for using brand names of software packages or hardware.

This document has **16** pages. Any blank pages are indicated.

Section A

You are advised to spend no longer than 40 minutes answering this section.

Here is a copy of the pre-release material.

DO NOT attempt Tasks 1, 2 and 3 now.

Use the pre-release material and your experience from attempting the tasks before the examination to answer Question 1.

Pre-release material

A program is needed for a quiz to help younger students to practise their multiplication tables. There needs to be two ways of using the quiz; testing and learning.

Testing: the student is given **one** attempt at answering each question and the score is calculated for the whole test.

Learning: the student is given up to **three** attempts to get their answer to each question correct. There is no scoring.

A student can choose which multiplication table, from 2 to 12, to use for the quiz. There are five questions in each quiz, each question must use the chosen multiplication table and a different whole number (from 1 to 12) as the multiplier.

Write and test a program or programs for a multiplication tables quiz.
- Your program or programs must include appropriate prompts for the entry of data; data must be validated on entry.
- Error messages and other output need to be set out clearly and understandably.
- All variables, constants and other identifiers must have meaningful names.

You will need to complete these **three** tasks. Each task must be fully tested.

Task 1 – Testing a student
Students enter their name and choice of multiplication table. Each question is displayed on the screen one at a time, for example:

> **Question 1**
>
> **2 X 7 =**

Students enter their answer and move on to the next question. A running total of correct answers (score) is kept. At the end of the quiz the student's name and score are displayed with a personalised message related to the score, for example:

> **Aarav your score is 5/5**
>
> **Well done full marks**

> **Diya your score is 3/5**
>
> **Have another practice**

Task 2 – Student learning
Students enter their name and choice of multiplication table. Each question is displayed on the screen as in **Task 1**. If an answer is correct, a personalised message containing the student's name confirms this, the quiz then moves to the next question. If an answer is incorrect, a personalised message containing the student's name and a hint is displayed, for example:

> **Aarav your answer is too large**

Up to three attempts are offered to get each answer correct. After the third incorrect attempt, the correct answer is displayed and the quiz moves on to the next question.

Task 3 – Varying the quiz
Modify **Task 1** to allow students to choose how many questions they would like in the test and if they would like a 'mixed' set of questions. A 'mixed' set means that each question can be from a different multiplication table; from 2 to 12.

1 All variables, constants and other identifiers must have meaningful names.

(a) Identify the variable that you used to store the student's answer in **Task 1**. Give the most appropriate data type for this variable. Explain how your program ensured that any data entered for the answer was valid.

Variable ....

Data type ....

Validation ....

....

....

....
[4]

(b) Identify and give the data type of a **different** variable, that you could have used in **Task 2**. State the use of this variable in **Task 2**.

Variable ....

Data type ....

Use ....

....
[3]

(c) Write an algorithm for **Task 1**, using **either** pseudocode, programming statements **or** a flowchart.

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

.... [6]

(d) Explain how your program completed **Task 3**.
Include any programming statements that you have added to **Task 1** and fully explain the purpose of each statement.

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

.... [4]

(e) Explain how you could alter **Task 1** to change the quiz to:

- display three alternative answers for each question
- allow the student to choose one of these answers

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

.... [3]

 [Turn over

Section B

2 An algorithm has been written in pseudocode to:

- input 25 positive whole numbers less than 100
- find and output the largest number
- find and output the average of all the numbers

```
01  A ← 0
O2  B ← 0
03  C ← 0
04  REPEAT
05    REPEAT
06      INPUT D
07    UNTIL D > 0 AND D < 100 AND D = INT(D)
08    IF D > B
09      THEN
10        B ← D
11    ENDIF
12    C ← C + D
13    A ← A + 1
14  UNTIL A >= 25
15  E ← C / A
16  OUTPUT "Largest number is ", B
17  OUTPUT "Average is ", E
```

(a) Give the line number for the statements showing:

Totalling ....

Counting ....

Range check ....

Calculating the average ....

[4]

(b) State an example for each type of test data needed to test the input of the number:

Normal test data example ....

Erroneous/abnormal test data example ....

Extreme test data example ....

[3]

(c) The algorithm needs to be changed to include finding and outputting the smallest number input. Describe how you would change the algorithm.

....

....

....

....

....

....

....

....

....

.... [4]

3 Four pseudocode statements and three flowchart symbols are shown.

Draw a line from each pseudocode statement to its correct flowchart symbol.

Pseudocode statement   **Flowchart symbol**

IF X > 12

INPUT X

X ← Y + Z

OUTPUT X

[4]

4   This algorithm accepts weights of bags of cookies. Any cookie bag weighing between 0.9 and 1.1 kilograms inclusive is acceptable. Underweight bags weigh less than 0.9 kilograms and overweight bags weigh more than 1.1 kilograms. An input of a negative number stops the process. Then the total number of bags, the number of overweight bags and the number of underweight bags weighed are output.

```
Accept ← 0
Over ← 0
Under ← 0
OUTPUT "Enter weight of first cookie bag"
INPUT BagWeight
WHILE BagWeight > 0
    IF BagWeight > 1.1
      THEN
        Error ← 1
      ELSE
        IF BagWeight < 0.9
          THEN
            Error ← 2
          ELSE
            Error ← 0
        ENDIF
    ENDIF
    CASE Error OF
      0 : Accept ← Accept + 1
      1 : Over ← Over + 1
      2 : Under ← Under + 1
    ENDCASE
  OUTPUT "Weight of next bag?"
  INPUT BagWeight
ENDWHILE
Total ← Accept – Over – Under
OUTPUT "Number of bags weighed ", Total
OUTPUT "Number overweight ", Over
OUTPUT "Number underweight ", Under
```

(a) Complete a trace table for the given algorithm using this input data:
1.05, 0.99, 1.2, 0.85, 1.1, 0.9, 1.5, 0.95, 1.05, 1.00, 1.07, 0.89, –10

BagWeight	Accept	Over	Under	Error	Total	OUTPUT

[7]

(b) There is an error in this algorithm.
Identify the error and write the corrected pseudocode statement.

Error ....

Correction ....

....

[2]

5 A database table, CHOCBAR, is used to keep a record of chocolate bars sold. Chocolate bars are categorised by:

- SIZE – small or large
- FILLING – brief description, for example mint crunch
- PRICE – price in Rupees, for example ₹2.50
- NUMBERSOLD – how many sold

A database management system uses these data types:

Text   Number   Currency   Boolean

(a) Select the most appropriate data type for these three fields from the four data types shown. Each data type must be different. State the reason why you chose the data type.

SIZE data type ....

Reason ....

....

PRICE data type ....

Reason ....

....

NUMBERSOLD data type ....

Reason ....

....
[3]

(b) Complete the query-by-example grid below to display only the price, filling and number sold of small chocolate bars that have sold fewer than 10 bars.

Field:				
Table:				
Sort:				
Show:	☐	☐	☐	☐
Criteria:				
or:				

[3]

BLANK PAGE

14

BLANK PAGE

15

BLANK PAGE

16

BLANK PAGE

Cambridge IGCSE™

COMPUTER SCIENCE	**0478/22**
Paper 2	**March 2021**
MARK SCHEME	
Maximum Mark: 50	

Published

This document consists of **8** printed pages.

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)	Variable `answer` (example only)
	Data type integer/int
	Validation two distinct different points **OR** one point and an expansion
	One mark per mark point, max **two**
	   – restricted input to positive numbers
	   – checked if input was numeric
	   – ensured that input was a whole number / integer
	   – checked if a value was input
	Or one point and an expansion **two** marks, for example
	   – restricted input to positive numbers (1) by using a range/limit check / REPEAT… UNTIL looping until value greater than zero (1)
	   – checked if input was numeric (1) by using a type check (1)
	   – ensured the value input was a whole number (1) any suitable method e.g. rounding to whole number
	   – checked if a value was input (1) by using a presence check (1)
		4

1(b)	Variable `noAttempts` (example only) Data type integer/int Use storing the number of attempts a pupil has made on a question	**3**

Question	Answer	Marks

1(c)	**One** mark per mark point, max **six**
	    MP1 initialisation (must include number of correct answers)
	    MP2 input name and table number with prompts
	    MP3 validate table number **and reinput**
	    MP4 ensure each question uses a different multiplier
	    MP5 display question
	    MP6 input **and validate** answer
	    MP7 check if answer correct
	    MP8 update score appropriately
	    MP9 repeat for 5 questions
	    MP10 display score **and name** at end …
	    MP11 …with an attempt to display an appropriate message dependent on score
	
	Example
	<pre>Score = 0
	Multiplier[1] ← 2
	Multiplier[2] ← 5
	Multiplier[3] ← 7
	Multiplier[4] ← 8
	Multiplier[5] ← 11
	OUTPUT "Multiplication Table Test"
	OUTPUT "Please enter your name"
	INPUT Name
	REPEAT
	    OUTPUT "Please enter the table (2 to 12) you would like
	to be tested"
	    INPUT Table
	UNTIL (Table >=2) AND (Table <=12)
	FOR Count = 1 TO 5
	    OUTPUT ("Question ", Count)
	    OUTPUT (Multiplier[Count], " X ", Table, " = ")
	    REPEAT
	        INPUT Answer
	    UNTIL Answer > 0 AND Answer = INT(Answer)
	    IF Answer = Multiplier[Count] * Table
	      THEN
	        Score ← Score + 1
	    ENDIF
	NEXT
	OUTPUT (Name, " your score is ", Score)
	IF Score = 5
	  THEN
	    OUTPUT ("Well done full marks")
	  ELSE
	    OUTPUT ("Have another practice")
	ENDIF</pre>
		6

Question	Answer	Marks

1(d)	Explanation
	One mark per mark point illustrated with a suitable programming statement, max **four**
	   MP1 extra prompt and input to choose the number of questions and/or mixed set
	   MP2 method of choosing number of questions and/or mixed set
	   MP3 extra prompt and input for number of questions
	   MP4 explanation of validation for either input
	   MP5 how the end value for the question loop was changed if required
	   MP6 how the program was changed to allow for mixed tables
	   MP7 … how the program ensured that the test used more than one multiplication table
		4

1(e)	Explanation
	One mark per mark point, max **three**
	   MP1 how to provide 3 answers, including the correct one
	   MP2 how to ensure that all 3 answers are different
	   MP3 how 3 answers are displayed e.g. output all 3 answers as a numbered list
	   MP4 how to select the correct answer e.g. input the number of the correct answer
	   MP5 how to ensure that the correct answer is not always in the same position
		3

Question	Answer	Marks
	Section B	
2(a)	– 12 – 13 – 07 – 15	**4**
2(b)	– 27 (example many correct answers any whole number 1-99 inclusive) – 106 (example many correct answers) – 99/1	**3**
2(c)	**One** mark per mark point, max **four** – **new** variable for minimum… – … set to first value/high value – … at start of program / before line 4 – test input / D for less than minimum – … replace value minimum if input less than – … after line 7 and before line 14 – new output for minimum (with appropriate message) – … at end of program // after line 14	**4**

Question	Answer	Marks
3	**Pseudocode statement**　　　　　**Flowchart symbol** IF X > 12 INPUT X X ← Y + Z OUTPUT X **One** mark for a single line to decision box from IF X > 12 **One** mark for a single line to process box from X ← Y + Z **Two** marks for two single lines from INPUT X and OUTPUT X to output box **OR** **One** mark for a single line from INPUT X or OUTPUT X to output box // two single lines from INPUT X and OUTPUT X to process box	**4**

Question	Answer	Marks
4(a)	**One** mark for each correct column	7

Bag Weight	Accept	Over	Under	Error	Total	OUTPUT
	0	0	0			(Enter weight of first cookie bag)
1.05	1			0		(Weight of next bag?)
0.99	2			0		(Weight of next bag?)
1.2		1		1		(Weight of next bag?)
0.85			1	2		(Weight of next bag?)
1.1	3			0		(Weight of next bag?)
0.9	4			0		(Weight of next bag?)
1.5		2		1		(Weight of next bag?)
0.95	5			0		(Weight of next bag?)
1.05	6			0		(Weight of next bag?)
1.00	7			0		(Weight of next bag?)
1.07	8			0		(Weight of next bag?)
0.89			2	2		(Weight of next bag?)
-10					4	
						Number of bags weighed 4
						Number overweight 2
						Number underweight 2

Question	Answer	Marks
4(b)	**Error** Total ← Accept - Over - Under // line 26 **Correction** Total ← Accept + Over + Under	2

Question	Answer	Marks
5(a)	**One** mark for data type and reason SIZE text, expressed as a single word // Boolean, only two choices PRICE currency, needs to be expressed as Rupees/money NUMBERSOLD number, integer values/could be used in calculations	3
5(b)		3

5(b)

Field:	SIZE	FILLING	PRICE	NUMBERSOLD
Table:	CHOCBAR	CHOCBAR	CHOCBAR	CHOCBAR
Sort:				
Show:	☐	☑	☑	☑
Criteria:	= "small"			<10
or:				

One mark for correct rows Field, Table and Sort
One mark for correct Show row
One mark for correct Criteria row