Cambridge Assessment International Education

Cambridge IGCSE™

CANDIDATE NAME									
CENTRE NUMBER						CANDIDATE NUMBER			

0671157253

COMPUTER SCIENCE **0478/21**

Paper 2 Problem-solving and Programming **October/November 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 following tasks before the examination to answer Question 1.

Pre-release material

A holiday park has eight squash courts that can be booked for an hour at a time. The first booking is from 08:00 to 09:00 and the last booking is from 17:00 to 18:00. All bookings start on the hour and bookings can only be made on the same day that the squash court is used. A screen displays today's date and how many squash courts are available for each hour.

When a booking is made, the name of the guest is recorded together with their mobile phone number. Once the squash court is booked, the guest is shown the court number together with a unique 4-digit code that can be used to unlock the squash court. Each booking is for one squash court for one hour. The 4-digit code must be different for each booking.

Write and test a program or programs for a computer system to manage the daily squash court bookings.

- 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 – at the start of the day

Write a program to set up arrays to record the following for each hour:
- whether a squash court is booked or available
- the name of the guest
- the mobile phone number of the guest
- the unique 4-digit code for the booking.

Set up a screen to display the court availability at the start of the day.

Task 2 – making a squash court booking

Check if there is a squash court available at the time requested. If a squash court is available, record the guest's name and mobile phone number. Mark the squash court as booked for that hour. Generate and record the unique 4-digit code for the booking. Display the mobile phone number for the guest to check, display the court number and the 4-digit code for the guest to remember. Display the updated court availability, showing an hour as fully booked if all the squash courts are now booked for that hour.

Task 3 – at the end of the day

Calculate the total number of squash court bookings. Find the hour(s) and court(s) with the most bookings. Display this information.

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

(a) Identify **one** constant that you could have used for **Task 1**. Give the value that would be assigned to this constant. State the use of this constant.

Constant .....

Value .....

Use .....

.....

[3]

(b) Describe the arrays that you have set up in **Task 1** to record today's data about the squash courts.

.....

.....

.....

.....

.....

.....

.....

..... [4]

(c) Explain how your program generates a unique 4-digit code for each booking.

.....

.....

.....

.....

.....

..... [3]

(d) Write an algorithm for **Task 2**. You do **not** need to display updated court availability. Use either pseudocode, programming statements or a flowchart. Assume that **Task 1** has been completed.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [6]

(e) Explain how your program completed the part of **Task 3** which calculates the total number of squash court bookings, finds the court(s) with the most bookings and displays this information. Include any programming statements that you have used and fully explain the purpose of each statement.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [4]

Section B starts on Page 8

Section B

2 An algorithm has been written in pseudocode to generate 50 positive random integers with values less than or equal to 100. These random integers are stored in the array `RandNum[ ]`

The function `Rand(X, Y)` generates a random integer greater than or equal to `X` and less than `Y`. For example, `Rand(1, 4)` generates `1` or `2` or `3`.

```
1 Count ← 0
2 REPEAT
3     RandNum[Counter] ← Rand(1, 100)
4     Count ← Count + 2
5 UNTIL Count <= 50
```

(a) Find the **four** errors in the pseudocode and write a correction for each error.

Error 1 .....

Correction .....

.....

Error 2 .....

Correction .....

.....

Error 3 .....

Correction .....

.....

Error 4 .....

Correction .....

.....

.....

[4]

(b) The pseudocode for this algorithm could be shortened by the use of a `FOR … NEXT` loop.

Rewrite the algorithm using a `FOR … NEXT` loop.

.....

.....

.....

..... [3]

(c) Identify another loop structure available in pseudocode.

..... [1]

3 A program has been written to check the value of a measurement. The measurement must be a positive number and given to three decimal places, for example, `3.982`

(a) (i) State suitable examples of normal and erroneous test data that could be used to test this program. For each example give the reason for your choice of test data.

Normal test data example .....

Reason .....

.....

Erroneous test data example .....

Reason .....

.....
[4]

(ii) Explain why two pieces of boundary test data are required for this program.
Give an example of each piece of boundary test data.

.....

.....

.....

.....

..... [3]

(b) Explain why verification is needed and how verification could be performed by this program.

.....

.....

.....

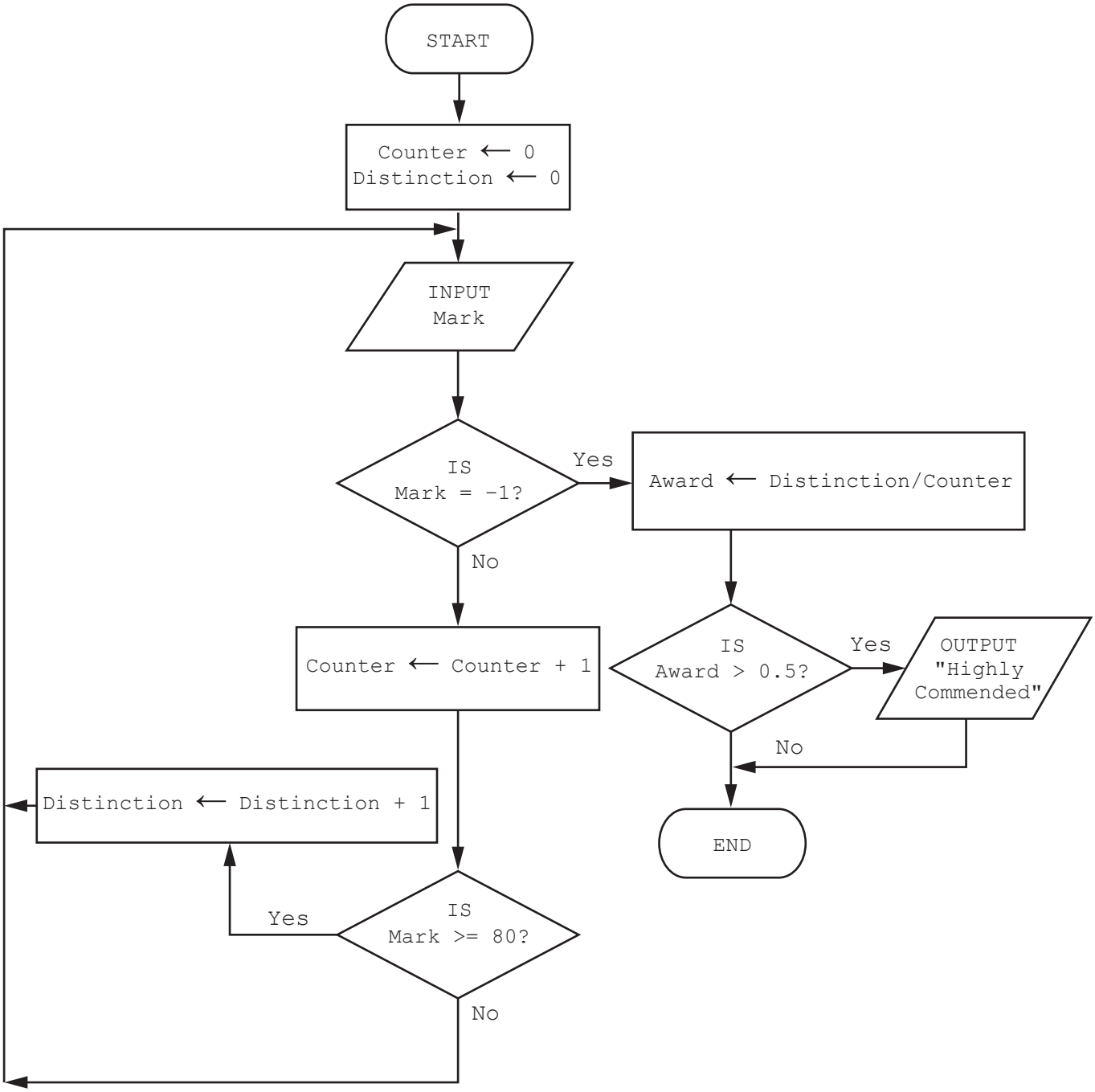
.....

.....

..... [3]

Question 4 starts on Page 12

4 The algorithm shown by this flowchart allows the input of examination marks for a class of students. A mark of –1 ends the process. If a mark is 80 or over then a distinction grade is awarded. The number of distinctions for the whole class is calculated. If this is over 50% of the class, the class is awarded a highly commended certificate.



Complete a trace table for the algorithm using this input data:
88, 74, 60, 90, 84, 87, 95, 72, 84, 66, –1

Counter	Distinction	Mark	Award	OUTPUT

[5]

5 A database table, APPLIANCE, is used to keep a record of kitchen appliances available for sale.

The following data is stored for each appliance:

- CATEGORY – washer, dishwasher, fridge or freezer
- ECONOMYRATING – A, B, C or D
- MANUFACTURER – Baku or ABC
- PRICE – price in $
- CODE – a unique code allocated by the manufacturer e.g. B982
- STOCK – number in stock.

The database management system uses these data types:

Text Number Currency Boolean

The ECONOMYRATING field and MANUFACTURER field have a data type of text.

(a) Identify the most appropriate data type for each field from the **four** types shown. State the reason why you chose each data type.

CATEGORY data type .....

Reason .....

.....

PRICE data type .....

Reason .....

.....

CODE data type .....

Reason .....

.....

STOCK data type .....

Reason .....

.....

[4]

(b) Complete the query-by-example grid to display only the category, manufacturer and code of the appliances with an economy rating of A.

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

[3]

BLANK PAGE

Cambridge Assessment International Education

Cambridge IGCSE™

COMPUTER SCIENCE **0478/21**

Paper 2 **October/November 2021**

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 2021 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.

Question	Answer	Marks
Section A		
1(a)	Constant `NoCourts // NoSessions // NoBookingSlots` Value 8 // 10 // 80 Use Storing the number of courts / sessions available / times and courts available for booking	**3**
1(b)	Any **four** from: Any **three** from MP1 Identifier / name of array used MP2 Description of purpose of an identified array MP3 Length of an identified array used MP4 Type of data in an identified array MP5 Explanation of number of arrays used, must be capable of storing all data required MP6 Sample data for an identified array **One mark** MP7 Identifying more than one array E.g.4 arrays of ten elements for each squash court, for example for squash court 1 `Availability1` of type Boolean, `Guest1`, `Mobile1` and `Code1` all type string	**4**
1(c)	Any **three** from: MP1 Setting the first code number MP2 How to find subsequent code numbers MP3 How to ensure they were unique MP4 How to ensure they were exactly 4 digits	**3**

Question	Answer	Marks

1(d)	Any **six** from:
	MP1 Input time
	MP2 … with prompt
	MP3 Check each court
	MP4 If court not booked …
	MP5 … set as booked
	MP6 Input name, mobile number
	MP7 Calculate unique 4-digit code
	MP8 Store name, mobile number and 4-digit code in an array
	MP9 Display mobile number and 4-digit code with suitable message
	MP10 Change mobile number if necessary
	
	Sample
	`OUTPUT ("Which time do you want to book a squash court")`
	`INPUT BookTime`
	`Booked ← FALSE`
	`CourtNumber ← 0`
	`IF BookTime >= 8 AND BookTime <= 17`
	`  THEN`
	`  BookTime ← BookTime - 7`
	`  IF Available1[BookTime] THEN Available1[BookTime] ← FALSE; CourtNumber ← 1; Booked ← TRUE ENDIF`
	`  IF NOT Booked AND Available2[BookTime] THEN Available2[BookTime] ← FALSE; CourtNumber ← 2; Booked ← TRUE ENDIF`
	`  IF NOT Booked AND Available3[BookTime] THEN Available3[BookTime] ← FALSE; CourtNumber ← 3; Booked ← TRUE ENDIF`
	`  IF NOT Booked AND Available4[BookTime] THEN Available4[BookTime] ← FALSE; CourtNumber ← 4; Booked ← TRUE ENDIF`
	`  IF NOT Booked AND Available5[BookTime] THEN Available5[BookTime] ← FALSE; CourtNumber ← 5; Booked ← TRUE ENDIF`
	`  IF NOT Booked AND Available6[BookTime] THEN Available6[BookTime] ← FALSE; CourtNumber ← 6; Booked ← TRUE ENDIF`
	`  IF NOT Booked AND Available7[BookTime] THEN Available7[BookTime] ← FALSE; CourtNumber ← 7; Booked ← TRUE ENDIF`
		6

Question	Answer	Marks

1(d)	IF NOT Booked AND Available8[BookTime] THEN Available8[BookTime] ← FALSE; CourtNumber ← 8; Booked ← TRUE ENDIF
	IF NOT Booked OUTPUT("No court available at ", BookTime + 7)
	ENDIF
	IF Booked
	THEN
	OUTPUT ("Please enter your name")
	INPUT Name
	Code ← Code + 1
	Correct ← "N"
	WHILE Correct = "N" DO
	OUTPUT ("Please enter your mobile number")
	INPUT Mobile
	OUTPUT ("Is Your mobile number correct Y/N? ", Mobile, " Your court number is ", CourtNumber, " Your entry code is ", Code)
	INPUT Correct
	ENDWHILE
	CASE CourtNumber OF
	1 : Guest1[BookTime}← Name; Mobile1[BookTime] ← Mobile; Code1[BookTime] ← Code
	2 : Guest2[BookTime}← Name; Mobile2[BookTime] ← Mobile; Code2[BookTime] ← Code
	3 : Guest3[BookTime}← Name; Mobile3[BookTime] ← Mobile; Code3[BookTime] ← Code
	4 : Guest4[BookTime}← Name; Mobile4[BookTime] ← Mobile; Code4[BookTime] ← Code
	5 : Guest5[BookTime}← Name; Mobile5[BookTime] ← Mobile; Code5[BookTime] ← Code
	6 : Guest6[BookTime}← Name; Mobile6[BookTime] ← Mobile; Code6[BookTime] ← Code
	7 : Guest6[BookTime}← Name; Mobile7[BookTime] ← Mobile; Code7[BookTime] ← Code
	8 : Guest8[BookTime}← Name; Mobile8[BookTime] ← Mobile; Code8[BookTime] ← Code
	ENDCASE
	NoCoutsBooked[BookTime] ← NoCoutsBooked[BookTime] - 1
	ELSE
	OUTPUT ("Court not booked”)
	ENDIF	

Question	Answer	Marks
1(e)	Explanation Any **four** from MP1 How the program totalled the number of bookings MP2 How the program displayed the total number of bookings MP3 How the program calculated the number of times each court was booked MP4 How the program selected the highest value for times a court was booked MP5 How the program displayed the court that was most booked Programming statements must be included and must be explained.	**4**

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td colspan="3">Section B</td></tr>
<tr><td>2(a)</td><td>Line 1 should be <code>Counter ← 0</code>
Line 3 <code>RandNum[Counter] ← Rand(1, 100)</code> should be <code>RandNum[Counter] ← Rand(1, 101)</code>
Line 4 <code>Counter ← Counter + 2</code> should be <code>Counter ← Counter + 1</code>
Line 5 <code>UNTIL Count <= 50</code> should be <code>UNTIL Counter >= 50 // UNTIL Counter = 50</code>

<pre>1 Counter ← 0
2 REPEAT
3   RandNum[Counter} ← Rand(1, 100)
4   Counter ← Counter + 1
5 UNTIL Counter >= 50</pre>Or

Line 3 <code>RandNum[Counter]</code> should be <code>RandNum[Count]</code>
Line 3 <code>Rand(1, 100)</code> should be <code>Rand(1, 101)</code>
Line 4 <code>Counter ← Counter + 2</code> should be <code>Count ← Count + 1</code>
Line 5 <code>UNTIL Count <= 50</code> should be <code>UNTIL Count >= 50 // UNTIL Count = 50</code>

<pre>1 Count ← 0
2 REPEAT
3   RandNum[Count} ← Rand(1, 100)
4   Count ← Count + 1
5 UNTIL Count >= 50</pre></td><td>4</td></tr>
<tr><td>2(b)</td><td>One mark for each correct line
<pre>FOR Count ← 0 TO 49 // FOR Count ← 1 TO 50
  RandNum[Count] ← Rand(1, 101) / Rand(0, 101)
NEXT // NEXT Count</pre></td><td>3</td></tr>
<tr><td>2(c)</td><td>Precondition loop <code>// WHILE … DO … ENDWHILE</code></td><td>1</td></tr>
</table>

Question	Answer	Marks
3(a)(i)	**one** mark for sample, **one** mark for reason **max four** Normal Sample any positive value with three decimal places e.g. 5.682 Reason to test that normal data is **accepted** and processed correctly Erroneous Sample any value that would be rejected e.g. 5.6 or -1.345 or seven Reason to test that erroneous data is **rejected**	**4**
3(a)(ii)	Reason to test that 0.000 / -0.001 / highest possible non-positive is rejected and 0.001 / 0.000 / lowest positive number is accepted Sample 1 0.000 Sample 2 0.001	**3**
3(b)	**One mark** To check that values are entered as intended // to prevent incorrect values that meet the validation criteria being accepted **Two marks** Asking the user to enter the value twice and comparing the values // double entry (1) only accepting a value if both entries are identical (1) or Displaying the value as it is entered (1) so the user can put right errors have been made as the value was entered (1)	**3**

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>4</td><td>One mark for each correct column

<table>
<tr><th>Counter</th><th>Distinction</th><th>Mark</th><th>Award</th><th>OUTPUT</th></tr>
<tr><td>0</td><td>0</td><td></td><td></td><td></td></tr>
<tr><td>1</td><td>1</td><td>88</td><td></td><td></td></tr>
<tr><td>2</td><td></td><td>74</td><td></td><td></td></tr>
<tr><td>3</td><td></td><td>60</td><td></td><td></td></tr>
<tr><td>4</td><td>2</td><td>90</td><td></td><td></td></tr>
<tr><td>5</td><td>3</td><td>84</td><td></td><td></td></tr>
<tr><td>6</td><td>4</td><td>87</td><td></td><td></td></tr>
<tr><td>7</td><td>5</td><td>95</td><td></td><td></td></tr>
<tr><td>8</td><td></td><td>72</td><td></td><td></td></tr>
<tr><td>9</td><td>6</td><td>84</td><td></td><td></td></tr>
<tr><td>10</td><td></td><td>66</td><td></td><td></td></tr>
<tr><td></td><td></td><td>-1</td><td>0.6</td><td>Highly Commended</td></tr>
</table></td><td>5</td></tr>
</table>

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>5(a)</td><td>CATEGORY – Text characters / words only used
PRICE – Currency, the price is in dollars / money
CODE – Text no calculations required, could be numbers or characters
STOCK – Number, comparisons and calculations may be required</td><td>4</td></tr>
<tr><td>5(b)</td><td>One mark for correct rows Field and Table
One mark for correct Show row
One mark for correct Criteria row

<table>
<tr><td>Field:</td><td>CATEGORY</td><td>MANUFACTURER</td><td>CODE</td><td>ECONOMYRATING</td></tr>
<tr><td>Table:</td><td>APPLIANCE</td><td>APPLIANCE</td><td>APPLIANCE</td><td>APPLIANCE</td></tr>
<tr><td>Sort:</td><td></td><td></td><td></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></td><td>="A"</td></tr>
<tr><td>or:</td><td></td><td></td><td></td><td></td></tr>
</table></td><td>3</td></tr>
</table>