Cambridge Assessment
International Education

Cambridge IGCSE™

CANDIDATE NAME	

CENTRE NUMBER		CANDIDATE NUMBER	

0405086665

COMPUTER SCIENCE **0478/23**

Paper 2 Problem-solving and Programming **October/November 2020**

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. 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 baguette ordering service allows customers to order filled baguettes. There are two sizes of baguette: 30 cm and 15 cm. Baguettes are available as white, brown or seeded bread. Baguettes have one filling and can have up to three salad items added.

Filling and salad choices are:

Filling	Salad
Beef Chicken Cheese Egg Tuna Turkey	Lettuce Tomato Sweetcorn Cucumber Peppers

Customers choose their baguette options. They then confirm their order, alter their choices or choose not to proceed.

Each day the ordering service calculates the number of each:
- size of baguette sold
- type of bread (white, brown or seeded) sold
- filling sold.

Write and test a program or programs for the baguette ordering service.
- 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 – Ordering.

Customers are given choices on size, type of bread, filling and salad items as shown. Only valid choices can be accepted. The customer is asked to confirm their order, alter their choices or choose not to proceed. If the customer confirms their order they are given a unique order number. Display the baguette ordered and the order number.

Task 2 – Recording the choices.

Extend TASK 1 to record totals for the size, types of bread and fillings sold that day and calculate the total number of baguettes sold that day.

Task 3 – Finding the most and least popular baguette fillings.

Using your results from TASK 2, display the most popular and least popular fillings as a percentage of the total number of baguettes sold that day.

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

(a) Identify **one** constant and identify **one** variable that you could have used for **Task 1**. Give the value that would be assigned to the constant. Give the data type for the variable. State what each one could be used for.

Constant ....

Value ....

Use ....

....

Variable ....

Data type ....

Use ....

.... [6]

(b) (i) Write an algorithm to allow a customer to choose the filling and salad items for their baguette (part of **Task 1**), using **either** pseudocode, programming statements **or** a flowchart. Your algorithm must only include this part of **Task 1**.

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

.... [5]

(ii) Explain how your algorithm in **part (b)(i)** ensured that only valid choices were accepted for the filling.

....

....

....

....

....

....

.... [3]

(c) Explain how you would need to change your program for **Task 1** if there were three sizes of baguette to choose from (15 cm, 20 cm and 30 cm).

....

....

....

.... [2]

(d) Explain how your program completed **Task 3**. Assume that **Task 2** has been completed. Any programming statements used in your answer must be fully explained.

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

.... [4]

Section B starts on Page 8.

Section B

2 An algorithm has been written in pseudocode to check the temperature readings taken from a freezer are within the range –18 degrees to –25 degrees inclusive.

The algorithm counts the number of times that the temperature reading is below –25 degrees and the number of times that the temperature reading is above –18 degrees.

An engineer is called if there are more than 10 temperature readings below –25 degrees.

An alarm sounds if there are more than 5 temperature readings above –18 degrees.

```
01  TooHot ← 0
02  TooCold ← 1000
03  REPEAT
04      OUTPUT "Please enter temperature"
05      INPUT Temperature
06      IF Temperature < -25
07        THEN
08          TooCold ← TooCold - 1
09      ENDIF
10      IF Temperature > -18
11        THEN
12          TooHot ← TooHot + 1
13      ENDIF
14  UNTIL TooHot > 5 OR TooCold > 10
15  IF TooHot < 5
16    THEN
17      INPUT "Alarm!!"
18  ENDIF
19  IF TooCold > 10
20    THEN
21      OUTPUT "Call the Engineer"
22  ENDIF
```

(a) Give the line number(s) from the algorithm of:

an assignment statement ....

a loop ....

a counting statement ....

a selection statement ....
[4]

(b) Give line numbers where the **four** errors are to be found in the pseudocode. Suggest a correction for each error.

Error 1 line number ....

Correction ....

....

Error 2 line number ....

Correction ....

....

Error 3 line number ....

Correction ....

....

Error 4 line number ....

Correction ....

....
[4]

(c) Explain how you could extend the algorithm to count the number of times the temperature readings are within the range –18 degrees to –25 degrees inclusive.

....

....

....

....

....

....

....

....

....

.... [4]

3 Four programming concepts and five descriptions are shown.

Draw a line to connect each **Programming concept** to its correct **Description**. Not all Descriptions will be connected to a Programming concept.

Programming concept

- Validation
- Verification
- Procedure
- Function

Description

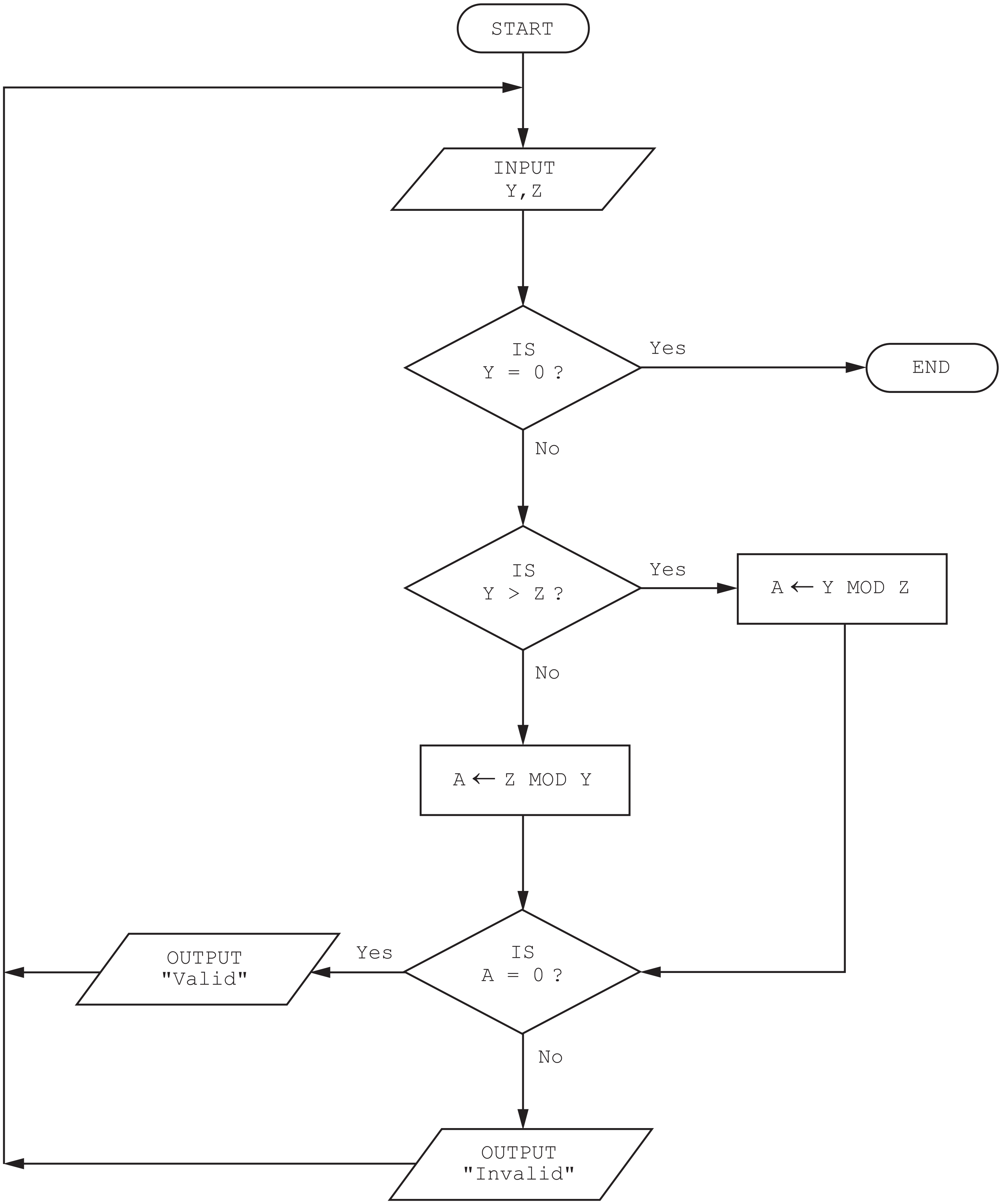
- A subroutine that does not have to return a value
- An automatic check to ensure that data input is reasonable and sensible
- A subroutine that always returns a value
- An overview of a program or subroutine
- A check to ensure that data input matches the original

[4]

Question 4 starts on Page 12.

4 This flowchart represents an algorithm that allows the input of two numbers and performs a calculation.

The predefined function MOD finds the remainder from integer division for example `X ← 8 MOD 5` gives `X` a value of `3`.



(a) Complete a trace table for this set of input data:
11, 4, 6, 2, 3, 9, 3, 2, 2, 6, 0, 0, 1, 1

Y	Z	A	OUTPUT

[4]

(b) Explain the purpose of this algorithm.

....

....

....

.... [2]

5 A marine wildlife rescue centre uses a database table, MARINE, to keep records of its creatures.

Creature	**Class**	**Quantity**	**Ready for release**	**Offspring**
Manta Ray	Fish	3	Y	N
Short-tailed Albatross	Bird	4	Y	N
Emperor Penguin	Bird	50	Y	Y
Bluefin Tuna	Fish	2	N	N
Manatee	Mammal	4	Y	N
Hawksbill Turtle	Reptile	10	Y	Y
Hammerhead Shark	Fish	3	Y	N
Yellow-eyed Penguin	Bird	4	Y	N
Kemp's Ridley Sea Turtle	Reptile	1	Y	N

(a) State how many fields and how many records are shown in this table.

Number of fields ....

Number of records .... [2]

(b) Show the output that would be given by this query-by-example.

Field:	Creature	Class	Ready for release
Table:	MARINE	MARINE	MARINE
Sort:			
Show:	☑	□	☑
Criteria:		="Bird"	
or:			

....

....

.... [2]

(c) Complete the query-by-example grid to display the creatures, in ascending order of quantity, that have no offspring and are ready for release. Display only the creature field.

Field:					
Table:					
Sort:					
Show:	□	□	□	□	□
Criteria:					
or:					

[4]

BLANK PAGE

Cambridge Assessment
International Education

Cambridge IGCSE™

COMPUTER SCIENCE **0478/23**

Paper 2 **October/November 2020**

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 2020 series for most Cambridge IGCSE™, Cambridge International A and AS Level and Cambridge Pre-U components, and some Cambridge O Level components.

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 name `MaxSalad` Value 3 Use Storing the salad items a baguette can have Variable name `OrderNumber` Data type integer Use storing the next order number available Many correct answers, this is an example only.	**6**

Question	Answer	Marks
1(b)(i)	Any **five** from: MP1 Display fillings available MP2 Prompt and entry of filling choice MP3 Only accept one correct filling MP4 Prompt and entry of number of salads MP5 Only accept 0 to 3 inclusive MP6 Display salads available MP7 Prompt and entry of correct number of salad choices MP8 Only accept correct salad choices MP9 At least one error message for incorrect data entry Example answer:	**5**

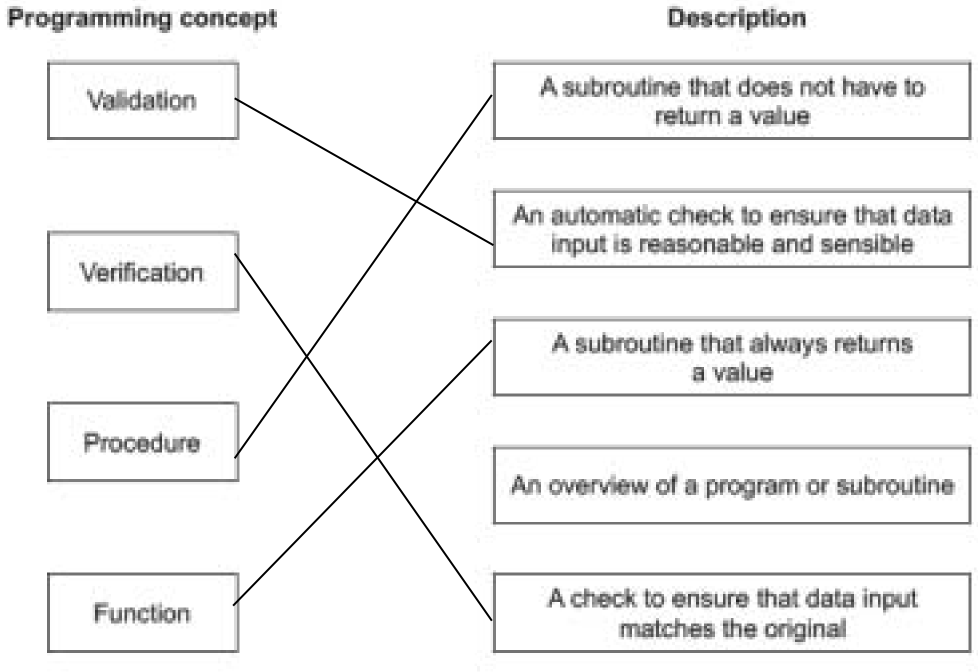
```
REPEAT
    PRINT "Beef        1"
    PRINT "Chicken     2"
    PRINT "Cheese      3"
    PRINT "Egg         4"
    PRINT "Tuna        5"
    PRINT "Turkey      6"
    OUTPUT "Please enter your choice of filling"
    INPUT Filling
UNTIL Filling >=1 AND Filling <=6
PRINT "Lettuce      1"
PRINT "Tomato       2"
PRINT "Sweetcorn    3"
PRINT "Cucumber     4"
PRINT "Peppers      5"
REPEAT
  PRINT "How many salads do you want"
  INPUT NumSalad
UNTIL NumSalad >= 0 and NumSalad <=3
WHILE NumSalad >0 DO
  PRINT "Enter Salad ", NumSalad
  OUTPUT "Please enter your choice of salad"
  INPUT SaladType
  CASE SaladType OF
    1: NumSalad ← NumSalad - 1
    2: NumSalad ← NumSalad - 1
    3: NumSalad ← NumSalad - 1
    4: NumSalad ← NumSalad - 1
    5: NumSalad ← NumSalad - 1
    OTHERWISE: PRINT "Error"
  ENDCASE
  Salad[NumSalad] ← SaladType

ENDWHILE
```

1(b)(ii)	Any **three** from: MP1 Provide a method of inputting the filling selection MP2 Check input is a correct filling e.g. using a `CASE` statement MP3 If not output a suitable error message MP4 If not provide a suitable method to re-input e.g. use of `REPEAT…UNTIL`	**3**

Question	Answer	Marks
1(c)	Any **two** from: MP1 Store three sizes of baguette e.g. add third baguette size to array MP2 Change the prompt to output three sizes of baguettes MP3 Change the selection statement to allow for a third size of baguette e.g. `IF/REPEAT/WHILE`…	**2**
1(d)	Explanation Any **four** from: MP1 check the total for **each** of the baguette fillings e.g. use of `FOR` loop MP2 use of two variables one for most popular filling and one for least popular filling MP3 method used to select the largest value as the most popular e.g. use of `IF` statement MP4 method used to select the smallest value as the least popular e.g. use of `IF` statement MP5 Use these values to calculate percentages MP6 … with the total number of baguettes sold MP7 Display results including suitable messages e.g. use of `PRINT` statement … Programming statements can be used but must be explained.	**4**

Question	Answer	Marks
Section B		
2(a)	Line 1/2/8/12 Line 3 and/or 14 Line 8/12 Line 6/10/15/19	**4**
2(b)	**One** mark for error and correction Line 02 `TooCold ← 0` Line 08 `TooCold ← TooCold + 1` Line 15 `IF TooHot > 5` Line 17 `OUTPUT "Alarm!!"`	**4**
2(c)	Any **four** from: Add a new variable `inRange` … … set to zero at start of algorithm Add extra `IF` statement `IF temperature >= -25 AND temperature <= -18` Update `inRange` by 1 if true	**4**

Question	Answer	Marks
3	**One** mark for each correct line  Programming concept Description Validation Verification Procedure Function A subroutine that does not have to return a value An automatic check to ensure that data input is reasonable and sensible A subroutine that always returns a value An overview of a program or subroutine A check to ensure that data input matches the original	4

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>4(a)</td><td><b>One</b> mark for each correct column
<table>
<tr><th>Y</th><th>Z</th><th>A</th><th>OUTPUT</th></tr>
<tr><td>11</td><td>4</td><td>3</td><td>Invalid</td></tr>
<tr><td>6</td><td>2</td><td>0</td><td>Valid</td></tr>
<tr><td>3</td><td>9</td><td>0</td><td>Valid</td></tr>
<tr><td>3</td><td>2</td><td>1</td><td>Invalid</td></tr>
<tr><td>2</td><td>6</td><td>0</td><td>Valid</td></tr>
<tr><td>0</td><td>0</td><td></td><td></td></tr>
</table></td><td>4</td></tr>
<tr><td>4(b)</td><td>Any <b>two</b> from:
Checking if the remainder, when the larger number is divided by the smaller number, is zero
To see if the larger number is a multiple of the smaller number
To see if the smaller number is a factor of the larger number</td><td>2</td></tr>
</table>

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>5(a)</td><td>5 fields
9 records</td><td><b>2</b></td></tr>
<tr><td>5(b)</td><td><b>One</b> mark for each correct column max <b>two,</b>
Or <b>two</b> correct rows <b>one</b> mark
Short-tailed Albatross Y
Emperor Penguin Y
Yellow-eyed Penguin Y</td><td><b>2</b></td></tr>
<tr><td>5(c)</td><td>
<table>
<tr><td>Field:</td><td>Creature</td><td>Quantity</td><td>Offspring</td><td>Ready for release</td></tr>
<tr><td>Table:</td><td>MARINE</td><td>MARINE</td><td>MARINE</td><td>MARINE</td></tr>
<tr><td>Sort:</td><td></td><td>Ascending</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>=N</td><td>=Y</td></tr>
<tr><td>or:</td><td></td><td></td><td></td><td></td></tr>
</table>
<b>One</b> mark for each correct field and table rows
<b>One</b> mark for sort row
<b>One</b> mark for show row
<b>One</b> mark for correct criteria</td><td><b>4</b></td></tr>
</table>