Cambridge Assessment International Education

Cambridge IGCSE™

CANDIDATE NAME			
CENTRE NUMBER		CANDIDATE NUMBER	

COMPUTER SCIENCE **0478/22**

Paper 2 Problem-solving and Programming **February/March 2022**

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 program is needed to record the number of strokes played by each player in a round of golf and decide who is the winning player.

The object of the game of golf is to hit a small ball into a series of small holes using a golf club. A score is kept of the number of times a player needs to hit the ball to get it in a hole; this is the number of strokes for the hole. The total score for a round of golf is the total number of strokes taken for each hole. The player with the least number of strokes is the winner.

A golf course consists of 9 or 18 holes. A round of golf is completed when all the holes have been played. There can be 2, 3 or 4 players taking part in a round of golf. The number of strokes that an experienced golfer would take to complete a round is called par. The scores for the round are displayed in relation to the par score. For example, if par for an 18-hole course was 72, a score of 80 would be 8 over par and a score of 70 would be 2 under par.

Write and test a program or programs to score a round of golf:

- Your program or programs must include appropriate prompts for the entry of data. Data must be validated on entry.
- All outputs, including error messages, 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 – setting up the round

Each player's scores for the round are to be stored in an array with a separate score for each hole. The scores in these arrays are to be set to zero before a round starts.

The following data is to be input and stored:

- the number of players taking part in the round
- the names of the players
- the number of holes to be played: 9 or 18
- the par for the course.

The number of players, their names, the number of holes to be played and the par for the course are all displayed to be checked before the round starts so that any errors seen can be corrected.

Task 2 – scoring the round

For each hole played, each player's name is displayed on the screen and they are asked to enter the number of strokes they played for that hole. Each player must enter the number of strokes twice to verify their score. A player can choose to see the total number of strokes that they have played so far in the round.

Task 3 – deciding the winning player

At the end of the round, display the name and final score for each player relative to par. The winner is identified; display their name and the winning score. There are also options that display:

- every player's score for each hole
- the player's name and hole number of any score of one for a hole (hole-in-one)
- the average score for the round
- the average score for a hole.

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

(a) State **one** constant and **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. Explain why the constant was used rather than a variable and explain why the variable was used rather than a constant.

Constant name ....

Value ....

Why a constant was used ....

....

....

Variable name ....

Data type ....

Why a variable was used ....

....

....
[6]

(b) Describe how your program set the scores to zero before each round for **Task 1**.

....

....

....

....

....

....

....

.... [4]

(c) Write an algorithm for **Task 2**, using pseudocode, programming statements or a flowchart.

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

[6]

(d) Explain how your program completed part of **Task 3** to identify the winner and to display their name and the winning score relative to par. All programming statements that you include must be fully explained.

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

.... [4]

Section B starts on page 8

Section B

2 An algorithm has been written to:

- input the ages of 100 students
- count and output the number of students aged 7 and under 12
- count and output the number of students aged 12 and under 18
- count and output the number of students aged 18 and over.

(a) Complete the pseudocode algorithm:

```
Count7to12 ← 0
Count12to18 ← 0
CountOver18 ← 0
FOR Student ← 1 TO ......................................
    OUTPUT "Please enter student's age in years "
    INPUT Age
    IF Age >= 7 .............................................
      THEN
        Count7to12 ← Count7to12 + 1
    ENDIF
    IF Age >= 12 AND Age < 18
      THEN
        Count12to18 ← ........................................
    ENDIF
    IF Age >= 18
      THEN
        CountOver18 ← CountOver18 + 1
    ENDIF
NEXT Student
OUTPUT "There are ", Count7to12, " students aged 7 and under 12."
OUTPUT "There are ", Count12to18, " students aged 12 and under 18."
OUTPUT "There are ", ................, " students aged 18 and over."
```

[4]

(b) Write the extra pseudocode statements that are needed to count and output the number of students under the age of 7. Use the variable `CountUnder7`; assume `CountUnder7` has already been set to zero.

....

....

....

....

....

....

.... [4]

3 **Four** validation checks and **five** descriptions are shown.

Draw a line from each validation check to the most appropriate description.

Validation check

length check

check digit

range check

type check

Description

checks that the data input is between two values

checks that the data input is an integer

checks that the data input has three digits

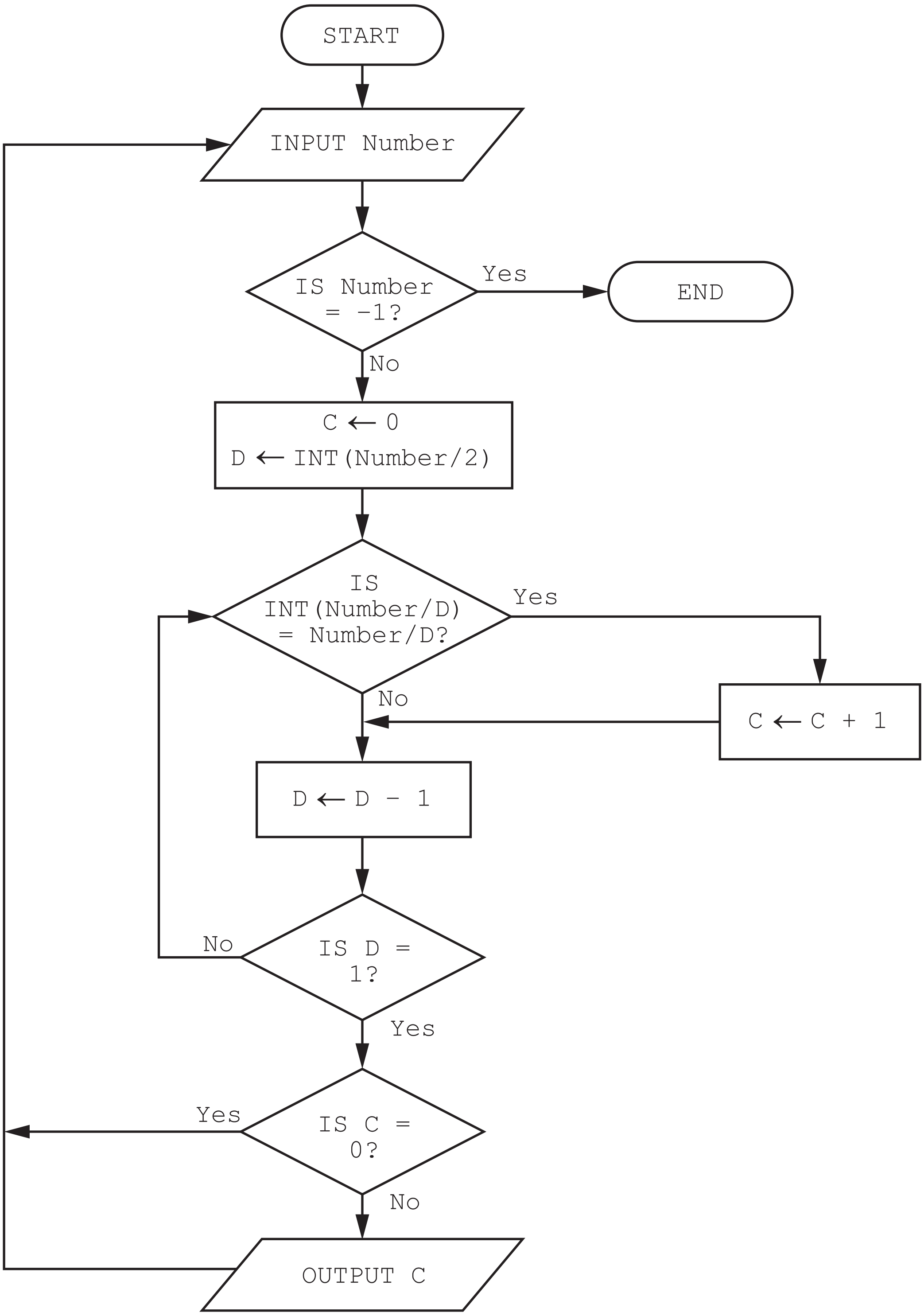
checks that the data has been input

checks that the data input has the correct digits

[4]

4 This flowchart inputs a whole number. The function INT returns the integer value of a number. For example, INT (7.5) is 7

An input of -1 ends the routine.



(a) Complete the trace table for the given algorithm using this input data:
7, 6, 5, 4, –1, 12, 34

`Number`	`C`	`D`	**OUTPUT**

[6]

(b) Describe the purpose of this algorithm.

....

....

....

.... [2]

(c) **(i)** Describe the problem that occurs if a whole number smaller than 4 and **not** equal to –1 is input.

....

....

....

.... [2]

(ii) Explain how to change the flowchart to prevent this problem occurring.

....

....

....

.... [2]

5 A database table, FLOWER, is used to keep a record of the type of flowers available to make up a bouquet.

FlowerID	Type	Colour	Style	Fragrance
CN001	Carnation	Pink	Stem	Y
CN002	Carnation	Red	Stem	N
CN103	Carnation	White	Stem	N
CN104	Carnation	Yellow	Stem	Y
CN105	Carnation	Pink	Spray	Y
CN106	Carnation	Red	Spray	N
CN107	Carnation	White	Spray	N
CN108	Carnation	Yellow	Spray	Y
RE101	Rose	Pink	Stem	Y
RE102	Rose	Red	Stem	Y
RE103	Rose	White	Stem	N
RE104	Rose	Yellow	Stem	Y
RE105	Rose	Orange	Spray	Y
RE106	Rose	Peach	Spray	N
LY101	Lily	White	Spray	Y

A query-by-example has been written to display just the type, style and colour of all flowers that have no fragrance.

Field:	FlowerID	Fragrance	Style	Colour
Table:	FLOWER	FLOWER	FLOWER	FLOWER
Sort:				
Show:	☑	☑	☑	☑
Criteria:		= Y		
or:				

Explain why the query-by-example is incorrect and write a correct query-by-example.

Explanation ....

....

....

....

....

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

[6]

BLANK PAGE

BLANK PAGE

BLANK PAGE

Cambridge IGCSE™

COMPUTER SCIENCE **0478/22**

Paper 2 **February/March 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 February/March 2022 series for most Cambridge IGCSE™, Cambridge International A and AS Level components and some Cambridge O Level components.

This document consists of **11** 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)	Many correct answers, the names used must be meaningful. The names given are examples only. **One** mark per mark point, max **three** Constant name `MaxNumberPlayers` Value 4 Why constant used This number will not change whilst the program is running **One** mark per mark point, max **three** Variable name `NumberOfPlayers` Data type Integer/int Why variable used A value between 2 and 4 inclusive is input and stored at the start of the round	**6**
1(b)	**One** mark per mark point, max **four** MP1 using a `FOR` … `NEXT` // `REPEAT` … `UNTIL` // `DO` … `WHILE` loop MP2 starting at 1/0 and finishing at 18/17 **or** 9/8 // 18/9 iterations MP3 setting the elements in the player scores arrays to zero MP4 … for all four arrays // number of arrays for players in this round MP5 setting variables / array for total scores to 0 MP6 use of assignment / append statement(s) Any programming statements included must be explained.	**4**

Question	Answer	Marks
1(c)	**One** mark per mark point, max **six**	**6**

MP1 loop through the number of holes played
MP2 for each hole work / loop through the number of players actually playing
MP3 for each player display their name
MP4 … prompt to enter the number of strokes played for the hole
MP5 … input the number of strokes twice
MP6 … validate both inputs are the same
MP7 … store the number of strokes in the appropriate array
MP8 … update the total score for that player
MP9 … prompt and input to ask if the player wants to see the number of strokes played so far
MP10 … check if required then output number of strokes

Example Answer

```
FOR Hole ← 1 TO NumberOfHoles
  FOR Player ← 1 TO NumberOfPlayers
    REPEAT
      OUTPUT PlayerName[Player], "Please enter the number of strokes played for hole ",
             Hole
      INPUT NumberStrokes
      OUTPUT PlayerName[Player], " please re-enter the number"
      INPUT NumberStrokesAgain
    UNTIL NumberStrokes = NumberStrokesAgain
    IF Player = 1
      THEN
        Player1[Hole] ← NumberStrokes
        Player1Total ← Player1Total + NumberStrokes
        OUTPUT "Do you want to see number total of strokes played so far Y/N? "
        INPUT SeeTotal
        IF SeeTotal = "Y"
          THEN
            OUTPUT "Total number of strokes so far ", Player1Total
        ENDIF
    ENDIF
    IF Player = 2
```

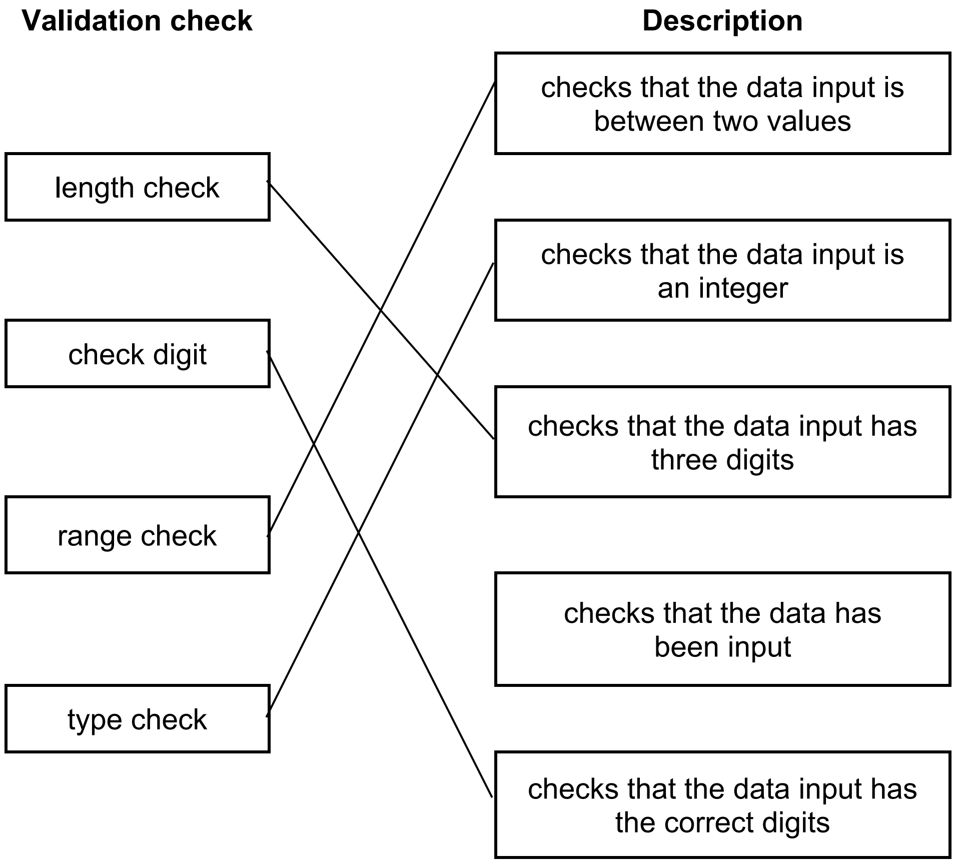
Question	Answer	Marks
1(c)		

```
      THEN
        Player2[Hole] ← NumberStrokes
        Player2Total ← Player2Total + NumberStrokes
        OUTPUT "Do you want to see number total of strokes played so far Y/N? "
        INPUT SeeTotal
        IF SeeTotal = "Y"
          THEN
            OUTPUT "Total number of strokes so far ", Player2Total
        ENDIF
    ENDIF
    IF Player = 3
      THEN
        Player3[Hole] ← NumberStrokes
        Player3Total ← Player3Total + NumberStrokes
        OUTPUT "Do you want to see number total of strokes played so far Y/N? "
        INPUT SeeTotal
        IF SeeTotal = "Y"
          THEN
            OUTPUT "Total number of strokes so far ", Player3Total
        ENDIF
    ENDIF
    IF Player = 4
      THEN
        Player4[Hole] ← NumberStrokes
        Player4Total ← Player4Total + NumberStrokes
        OUTPUT "Do you want to see number total of strokes played so far Y/N? "
        INPUT SeeTotal
        IF SeeTotal = "Y"
          THEN
            OUTPUT "Total number of strokes so far ", Player4Total
        ENDIF
    ENDIF
  NEXT Player
NEXT Hole
```

Question	Answer	Marks
1(d)	Explanation **One** mark per mark point, max **four** MP1 Work/ Loop through all the total scores MP2 compare each total score using selection / `IF` statements // use an appropriate function MP3 select the score with the **lowest** value MP4 … also select the player name for that score MP5 output the player name and **either** the difference between par and their score **or** their score MP6 compare the score with the lowest value with the value of par using selection / `IF` statements MP7 … identify as "over par" if the value is greater than par or output "under par" if the value is less than par or output "par" if there is no difference Any programming statements included must be explained.	**4**

Question	Answer	Marks
Section B		
2(a)	**One** mark per mark point, max **four** • `100` • `AND Age < 12` • `Count12to18 + 1` • `CountOver18`	**4**

2(b)	**One** mark suitable IF construct, **one** mark correct assignment statement, for example
	`IF Age < 7`
	`  THEN`
	`    CountUnder7 ← CountUnder7 + 1`
	`ENDIF`
	One mark suitable message, **one** mark correct use of `countUnder7` variable, for example
	`OUTPUT "There are ", CountUnder7, " students aged under 7."`
		4

Question	Answer	Marks
3	**One** mark for each correct **single** line from the validation check, max **four**  **Validation check** length check check digit range check type check **Description** checks that the data input is between two values checks that the data input is an integer checks that the data input has three digits checks that the data has been input checks that the data input has the correct digits	**4**

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>4(a)</td><td><b>One</b> mark each for columns <code>Number</code> and <b>OUTPUT</b>
<b>Two</b> marks for column <code>C</code> first four values (1) last three values (1)
<b>Two</b> marks for column <code>D</code> first six values (1) last four values (1)
<table>
<tr><th>Number</th><th>C</th><th>D</th><th>OUTPUT</th></tr>
<tr><td>7</td><td>0</td><td>3</td><td></td></tr>
<tr><td></td><td></td><td>2</td><td></td></tr>
<tr><td></td><td></td><td>1</td><td></td></tr>
<tr><td>6</td><td>0</td><td>3</td><td></td></tr>
<tr><td></td><td>1</td><td>2</td><td></td></tr>
<tr><td></td><td>2</td><td>1</td><td>2</td></tr>
<tr><td>5</td><td>0</td><td>2</td><td></td></tr>
<tr><td></td><td></td><td>1</td><td></td></tr>
<tr><td>4</td><td>0</td><td>2</td><td></td></tr>
<tr><td></td><td>1</td><td>1</td><td>1</td></tr>
<tr><td>-1</td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
</table>
</td><td><b>6</b></td></tr>
<tr><td>4(b)</td><td><b>One</b> mark per mark point, max <b>two</b>
• to <b>count</b> the factors / the numbers that go into (other than 1 or itself) of a number
• to output the <b>number of factors</b></td><td><b>2</b></td></tr>
</table>

Question	Answer	Marks
4(c)(i)	**One** mark per mark point, max **two** • the value of D becomes zero • division by zero error • endless loop	2
4(c)(ii)	**One** mark per mark point, max **two** • after the decision box to test if the number is -1 • insert another decision box to test if the number is less than 4 / less than or equal to 3 • return to INPUT Number if true	2

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>5</td><td>Explanation
<b>One</b> mark per mark point, max <b>three</b>
• field, FlowerID, not required / should not be displayed
• Type field not included and displayed
• Fragrance field should not be displayed
• Fragrance criteria should not be Y / should be N
<table>
<tr><td>Field:</td><td>Type</td><td>Fragrance</td><td>Style</td><td>Colour</td></tr>
<tr><td>Table:</td><td>FLOWER</td><td>FLOWER</td><td>FLOWER</td><td>FLOWER</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>= N</td><td></td><td></td></tr>
<tr><td>or:</td><td></td><td></td><td></td><td></td></tr>
</table>
query-by-example grid
<b>One</b> mark per mark point, max <b>three</b>

• One mark for changing Flower ID to Type
• One mark for changing Criteria in Fragrance to N
• One mark for changing Show in Fragrance to □</td><td>6</td></tr>
</table>