Exercise walk-through

11.2.4

Trace tables: dry-running an algorithm and finding logic errors

A-Level Computer Science

You must be able to

- complete a trace table for an algorithm that contains selection and loops
- state the output and the purpose of an algorithm from its trace table
- use a trace table to find a logic error, and correct the error

1 The method

Method — How to complete a trace table

1. Make one column for **each variable**, and one column for **OUTPUT**. Put the columns in the order in which the variables first appear.
2. Follow the pseudocode **one line at a time**, exactly as it is written, not as you think it should work.
3. When a variable **changes**, write the new value in its column. Do not write a value again if it did not change.
4. Go down to a **new row** for each pass of a loop (or when the next value in a column is already filled).
5. When a condition is tested, work it out with the values in the table **at that moment**.
6. Do not forget the **last test** of a loop: the one that is FALSE and ends it.

Method — A worked trace

Trace this algorithm for the inputs 4, 7, 2 and 0.

Method — A worked trace

```
Count ← 0
Big ← 0
INPUT N
WHILE N <> 0
    Count ← Count + 1
    IF N > Big THEN
        Big ← N
    ENDIF
    INPUT N
ENDWHILE
OUTPUT Count, " ", Big
```

Method — A worked trace

N	Count	Big	OUTPUT
	0	0	
4	1	4	
7	2	7	
2	3		
0			3 7

- Row 4: `2 > 7` is FALSE, so `Big` does not change and its cell stays empty.
- Row 5: `N` is 0, so the test `N <> 0` is FALSE, the loop ends, and the output is written.
- **The purpose:** the algorithm counts the numbers that are input before the 0, and finds the largest of them.

Method — Finding a logic error with a trace table

A **logic error** is an error in the design of the algorithm: the program runs, but gives a wrong result. A trace table shows **where the values first go wrong**.

```
Total ← 0
FOR Index ← 1 TO 4
    Total ← Index
NEXT Index
OUTPUT Total
```

Method — Finding a logic error with a trace table

Index	Total	OUTPUT
	0	
1	1	
2	2	
3	3	
4	4	4

Method — Finding a logic error with a trace table

`Total` is **replaced** on each pass, not added to: the second row should be 3, but it is 2. The error is the line `Total ← Index`; the correction is `Total ← Total + Index`.

In an exam answer give three things: the **line** that is wrong, **what it does** wrong (use values from your table), and the **corrected line**.

2 Practice

Practice 2.1 ▪ 3 marks

Complete the trace table for this algorithm when the input is 12.

```
INPUT Age
Price ← 10
IF Age < 16 THEN
    Price ← Price - 4
ENDIF
Price ← Price * 2
OUTPUT Price
```

Practice 2.1 ▪ 3 marks

Age	Price	OUTPUT

Solution 2.1

Method: A worked trace — p.5

Worked steps

- `Age` is 12, and `Price` starts at 10 **B1**
- `12 < 16` is TRUE, so `Price` becomes $10 - 4 = 6$ **B1**
- `Price` becomes $6 \times 2 = 12$, and **12** is output **B1**

Solution 2.1

Age	Price	OUTPUT
12	10	
	6	
	12	12

Practice 2.2 ▪ 4 marks

Complete the **trace table** for this algorithm when the input X is 5.

```
A ← 1
B ← 1
WHILE A < X
    B ← B * 2
    A ← A + 1
ENDWHILE
OUTPUT B
```

Practice 2.2 ▪ 4 marks

A	B	OUTPUT
1	1	

Solution 2.2

Method: A worked trace — p.5

Worked steps

- B doubles on each pass until A reaches 5 M1 M1

Solution 2.2

A	B	OUTPUT
1	1	
2	2	
3	4	
4	8	
5	16	16

- stops when $A = 5$; OUTPUT **16** A1 A1

Practice 2.3 ▪ 4 marks

Study this pseudocode.

```
Value ← 1
REPEAT
    Value ← Value + 3
    OUTPUT Value
UNTIL Value > 9
```

Practice 2.3 (a) ▪ 3 marks

Complete the trace table.

Value	OUTPUT
1	

Solution 2.3 (a)

Method: How to complete a trace table — p.4

Worked steps

one mark for each correct row after the first:

Value	OUTPUT
1	
4	4
7	7
10	10

Practice 2.3 (b) ▪ 1 mark

State how many times the body of the loop runs.

Solution 2.3 (b)

Worked steps

`10 > 9` is TRUE after the third pass, so the body runs **3** times B1

Practice 2.4 ▪ 3 marks

Complete the trace table for this algorithm.

```
Total ← 0
FOR Outer ← 1 TO 3
    FOR Inner ← 1 TO Outer
        Total ← Total + Inner
    NEXT Inner
NEXT Outer
OUTPUT Total
```

Practice 2.4 ▪ 3 marks

Outer	Inner	Total	OUTPUT
		0	
1	1	1	

Solution 2.4

Method: Finding a logic error with a trace table — p.8

Worked steps

Solution 2.4

Outer	Inner	Total	OUTPUT
		0	
1	1	1	
2	1	2	
	2	4	
3	1	5	
	2	7	
	3	10	
			10

- `Outer` and `Inner` columns correct **B1**; `Total` column correct **B1**; output **10** **B1**

3 Exam-style

Exam-style 3.1 ▪ 9 marks

Study this pseudocode.

Exam-style 3.1 ▪ 9 marks

```
Count ← 0
Total ← 0
INPUT Value
WHILE Value <> 999
    IF Value > 0 THEN
        Total ← Total + Value
        Count ← Count + 1
    ENDIF
    INPUT Value
ENDWHILE
IF Count > 0 THEN
    OUTPUT Total / Count
ELSE
    OUTPUT "No data"
ENDIF
```

Exam-style 3.1 (a) ▪ 4 marks

Complete the trace table for the inputs 6, -2, 9, 0 and 999.

Value	Count	Total	OUTPUT
	0	0	

Solution 3.1 (a)

Method: A worked trace — p.5

Worked steps

Solution 3.1 (a)

Value	Count	Total	OUTPUT
	0	0	
6	1	6	
-2			
9	2	15	
0			
999			7.5

- `Value` column B1; `Count` column B1; `Total` column B1; output **7.5** B1

Exam-style 3.1 (b) ▪ 2 marks

State the purpose of the algorithm.

Solution 3.1 (b)

Worked steps

it outputs the mean (average) of the **positive** values input B1

- it stops at the rogue value 999, and outputs `"No data"` if there were no positive values B1

Exam-style 3.1 (c) ▪ 3 marks

The algorithm is rewritten with a post-condition loop, and `INPUT Value` becomes the first statement inside the loop. **Explain** what goes wrong if the first value input is 999, and how to correct it.

Solution 3.1 (c)

Worked steps

the test comes at the end of the loop, so 999 is processed before the loop stops B1

- 999 is positive, so it is added to `Total` and counted, and the output is 999 instead of `"No data"` B1
- correct it by changing the condition to `Value > 0 AND Value <> 999` B1

Exam-style 3.2 ▪ 8 marks

This algorithm should output the **largest** of three numbers that are input.

```
Largest ← 0
FOR Index ← 1 TO 3
    INPUT Number
    IF Number > Largest THEN
        Largest ← Number
    ENDIF
NEXT Index
OUTPUT Largest
```

Exam-style 3.2 (a) ▪ 4 marks

Complete the trace table for the inputs -4, -9 and -2.

Index	Number	Largest	OUTPUT
		0	

Solution 3.2 (a)

Method: Finding a logic error with a trace table — p.8

Worked steps

one mark for each correct row:

Index	Number	Largest	OUTPUT
		0	
1	-4		
2	-9		
3	-2		0

Solution 3.2 (a)

- no input is greater than 0, so `Largest` never changes

Exam-style 3.2 (b) ▪ 1 mark

State the value that **should** be output for these inputs.

Solution 3.2 (b)

Worked steps

-2 B1

Exam-style 3.2 (c) ▪ 3 marks

Explain the logic error, and describe how to correct the algorithm.

Solution 3.2 (c)

Worked steps

`Largest` starts at 0, and 0 is greater than every number in this list, so no number ever replaces it B1

- the starting value must not be a fixed number: it must be the **first number** that is input B1
- for example: input the first number into `Largest` before the loop and run the loop for the other two numbers; or, inside the loop, `IF Index = 1 THEN Largest ← Number` B1

Exam-style 3.3 ▪ 7 marks

Study this pseudocode. The function `LENGTH()` returns the number of characters in a string, and `MID(Word, Position, 1)` returns the one character at `Position`.

```
Word ← "CAT"
NewWord ← ""
FOR Position ← LENGTH(Word) TO 1 STEP -1
    NewWord ← NewWord & MID(Word, Position, 1)
NEXT Position
OUTPUT NewWord
```

Exam-style 3.3 (a) ▪ 4 marks

Complete the trace table.

Position	NewWord	OUTPUT
	""	

Solution 3.3 (a)

> Method: How to complete a trace table — p.4

Worked steps

one mark for each correct row after the first, and one for the output:

Position	NewWord	OUTPUT
	`""`	
3	`"T"`	
2	`"TA"`	
1	`"TAC"`	TAC

Exam-style 3.3 (b) ▪ 1 mark

State the purpose of the algorithm.

Solution 3.3 (b)

Worked steps

it **reverses** the string B1

Exam-style 3.3 (c) ▪ 2 marks

A student changes the first line of the loop to `FOR Position ← 1 TO LENGTH(Word)`. State what is now output, and explain why.

Solution 3.3 (c)

Worked steps

CAT is output B1

- the characters are now taken from position 1 to position 3, so they are joined in their original order B1

Exam-style 3.4 ▪ 10 marks

A shop wants an algorithm that inputs the prices of 4 items and outputs their mean. A student writes:

```
Count ← 0
Total ← 0
WHILE Count <= 4
    INPUT Price
    Total ← Total + Price
    Count ← Count + 1
ENDWHILE
OUTPUT Total / 4
```

Exam-style 3.4 (a) ▪ 5 marks

Complete the trace table. The prices 2, 4, 6, 8 and 10 are available as input, in this order. Use only as many as the algorithm asks for.

Count	Price	Total	OUTPUT
0		0	

Solution 3.4 (a)

> Method: A worked trace — p.5

Worked steps

one mark for each correct row after the first:

Solution 3.4 (a)

Count	Price	Total	OUTPUT
0		0	
1	2	2	
2	4	6	
3	6	12	
4	8	20	
5	10	30	7.5

- when `Count` is 4, the test `4 <= 4` is still TRUE, so the body runs a fifth time

Exam-style 3.4 (b) ▪ 1 mark

State the number of prices that the algorithm inputs.

Solution 3.4 (b)

Worked steps

5 B1

Exam-style 3.4 (c) ▪ 2 marks

Identify the line that contains the error, and write the corrected line.

Solution 3.4 (c)

Worked steps

the line `WHILE Count <= 4` B1

- `WHILE Count < 4` B1

Exam-style 3.4 (d) ▪ 2 marks

State the type of loop that would make this error less likely, and give a reason.

Solution 3.4 (d)

Worked steps

a **count-controlled (`FOR`)** loop B1

- `FOR Count ← 1 TO 4` states the number of passes directly, so the programmer does not have to write the test and the increase of the counter B1