ANSWERS

F5.4 Trace tables

Algorithms · GCSE · OCR J277 2.1.2, AQA 8525 3.1.1, Edexcel 1CP2 1.2.4 · about 15 min

What this lesson is about

Following an algorithm line by line, and tracing to find a logic error.

- A first trace
- Let the program trace itself
- Tracing to find a bug
- Tracing a robot

Questions

5 marks in all

1. What does this program print? [1 mark]

```
total = 0
for i in range(1, 5):
    total = total + i * 2
print(total)
```

Answer:

```
20
```

total goes 2, 6, 12, 20.

2. What does this program print? [1 mark]

```
x = 1
y = 10
while x < y:
    x = x * 2
    y = y - 1
print(x, y)
```

Answer:

```
8 7
```

x: 2, 4, 8; y: 9, 8, 7. When x is 8 and y is 7, x < y is False.

3. A program prints the average of five readings as 4.0 when it should be 30.0. A trace shows total goes 40, 35, 30, 25, 20. What is the bug? [1 mark]

- ☐ **A** total = 0 is inside the loop, so it resets each time
- ☐ **B** The readings are wrong
- ☐ **C** The loop runs too many times
- ☐ **D** Division is broken

Answer: A. The total never grows because it is set back to 0 every time round.

4. What is a trace table used for? [1 mark]

- ☐ **A** Recording variable values step by step to check what an algorithm does
- ☐ **B** Storing a program's data
- ☐ **C** Drawing a flowchart
- ☐ **D** Timing a program

> **Answer:** A. It shows exactly how values change, which reveals the output and any logic errors.

5. What does this program print? [1 mark]

```
n = 20
steps = 0
while n > 1:
    n = n // 2
    steps = steps + 1
print(n, steps)
```

> **Answer:**
>
> ```
> 1 4
> ```
>
> n goes 10, 5, 2, 1, so there are four steps.

The task: trace and fix

Fix the average program so it prints `average: 30.0`. Keep a trace: inside the loop, print a line `i=<i> total=<total>` each time round, so the program shows its own trace table.

```
# the two lines every program starts with: the commands, then the robot
from bugbot import *
connect()

readings = [40, 35, 30, 25, 20]
for i in range(len(readings)):
    total = 0
    total = total + readings[i]
print("average:", total / len(readings))
```

The hint students can ask for: Trace what the total does each time round. One line sets it back to its starting value when it should not. Move that line so it runs once, and print the counter and the total each time round so you can see the fix work.

A solution

```
from bugbot import *
connect()
readings = [40, 35, 30, 25, 20]
total = 0
for i in range(len(readings)):
    total = total + readings[i]
    print(f"i={i} total={total}")
print("average:", total / len(readings))
```

Any program that meets the task's checks is marked correct in the simulator; this is one way, not the only way.