Solutions

Each answer shows the steps, then the **result**.

2.1

(a) $20 - 1 + 1 =$ **20** times

(b) $9 - 0 + 1 =$ **10** times

(c) the counter takes the values 5, 10, 15, 20, 25: **5** times

2.2

(a) the counter is 3, 4, 5, 6, and each is doubled: **6, 8, 10, 12**

(b) the counter goes down by 3 each time: **9, 6, 3**

2.3

- `FOR … STEP 2` from 2 to 20

```
FOR i ← 2 TO 20 STEP 2
    OUTPUT i
NEXT i
```

- *(outputting `i` only when `i MOD 2 = 0` also scores)*

2.4

- (1) the total starts at `0`
- (2) each value is added: **`Total + Value`**
- (3) the counter of this loop: **`Index`**
- (4) the mean of 10 values: **`Total / 10`**

2.5

- `Total ← 0` is **inside** the loop, so the total goes back to 0 on every pass, and only the last price is left
- move `Total ← 0` to the line **before** `FOR`

2.6

- input the first height and store it as the smallest
- a loop that runs for the other 24 heights
- compare each height with the smallest, and replace it if smaller
- output the smallest after the loop


```
DECLARE Count : INTEGER
DECLARE Height, Smallest : REAL
INPUT Height
Smallest ← Height
FOR Count ← 2 TO 25
    INPUT Height
    IF Height < Smallest THEN
        Smallest ← Height
    ENDIF
NEXT Count
OUTPUT Smallest
```

3.1

- the variables declared, and `INPUT Table`
- a `FOR` loop from 1 to 12, closed with `NEXT`
- the product `Line * Table` worked out inside the loop
- the `OUTPUT` has the values and the text in the correct order
- the text `" x "` and `" = "` is in quotes, with commas between the items

```
DECLARE Table, Line : INTEGER
INPUT Table
FOR Line ← 1 TO 12
    OUTPUT Line, " x ", Table, " = ", Line * Table
NEXT Line
```

3.2

- the variables declared with suitable types
- the total and the count set to 0 before the loop
- a `FOR` loop that runs 40 times, with the input inside it
- each time added to the total
- an `IF` that adds 1 to the count when the time is less than 60
- the count and `Total / 40` output **after** the loop

```
DECLARE Swimmer, Fast : INTEGER
DECLARE Time, Total : REAL
Total ← 0
Fast ← 0
FOR Swimmer ← 1 TO 40
    INPUT Time
    Total ← Total + Time
    IF Time < 60 THEN
        Fast ← Fast + 1
    ENDIF
NEXT Swimmer
OUTPUT Fast
OUTPUT Total / 40
```

3.3

- declarations; a count-controlled loop from 1 to 30; input inside the loop
- count the days with rainfall 0; running total
- keep the largest rainfall and its day number, replacing only when greater
- output all three after the loop

```
DECLARE Day, DryDays, WettestDay : INTEGER
DECLARE Rain, Total, MostRain : REAL
DryDays ← 0
Total ← 0
MostRain ← -1
WettestDay ← 0
FOR Day ← 1 TO 30
    INPUT Rain
    IF Rain = 0 THEN
        DryDays ← DryDays + 1
    ENDIF
    Total ← Total + Rain
    IF Rain > MostRain THEN
        MostRain ← Rain
        WettestDay ← Day
    ENDIF
NEXT Day
OUTPUT DryDays
OUTPUT Total
OUTPUT WettestDay
```

- `MostRain` starts at -1, below any real rainfall, so day 1 is always stored first; `>` rather than `>=` keeps the earlier day

3.4

(a) the inner loop runs 4 times for each of the 3 passes of the outer loop: $3 \times 4 = \mathbf{12}$ values

(b) the first pass of the outer loop gives 1×1 to 1×4, and the second pass starts with 2×1

- **1, 2, 3, 4, 2**

(c) `Line` is set to an empty string at the start of **each** pass of the outer loop

- inside the inner loop, the product is changed to a string and joined to `Line` with `&`
- a space is joined between the values
- `OUTPUT Line` comes after `NEXT Inner` and before `NEXT Outer`


```
FOR Outer ← 1 TO 3
    Line ← ""
    FOR Inner ← 1 TO 4
        Line ← Line & NUM_TO_STR(Outer * Inner) & " "
    NEXT Inner
    OUTPUT Line
NEXT Outer
```