Solutions

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

2.1

- `Ahmed` → `Name STRING`; `16` → `Age INTEGER`; `72.5` → `Average REAL`; `TRUE` → `Passed BOOLEAN`

2.2

- start with a few **high-level steps**
- then **break each step down** (refine it) into smaller steps, until every step is simple enough to write as code

2.3

- declare the counters, then loop 20 times with the test inside

```
DECLARE Index  : INTEGER
DECLARE Number : INTEGER
DECLARE Big    : INTEGER
Big ← 0
FOR Index ← 1 TO 20
    INPUT Number
    IF Number > 100 THEN
        Big ← Big + 1
    ENDIF
NEXT Index
OUTPUT Big
```

- `IF Number > 100` then add 1; `OUTPUT` after the loop

2.4

- **iteration** — the `FOR` loop repeats the input-and-test 20 times
- **or** selection — the `IF` decides whether to add 1

2.5

- (1) `Total + Value`
- (2) `Total / 10`
- (3) `"Average = ", Average`

3.1

- declare and generate `Secret`; loop until the guess matches


```
DECLARE Secret : INTEGER
DECLARE Guess  : INTEGER
Secret ← RANDOM(1, 50)
REPEAT
    INPUT Guess
    IF Guess < Secret THEN
        OUTPUT "higher"
    ENDIF
    IF Guess > Secret THEN
        OUTPUT "lower"
    ENDIF
UNTIL Guess = Secret
OUTPUT "correct"
```

- `INPUT` inside the loop; too-low → `"higher"`; too-high → `"lower"`
- loop ends when `Guess = Secret`; `"correct"` once after the loop
- *(a `WHILE` loop with a first input before it is equally valid)*

3.2

- (1) `Count ← 1` (set the array index to the first element)
- (2) decision `Value = -1?` (the sentinel that ends the loop)
- (3) `Data[Count] ← Value` then `Count ← Count + 1` (store, then move to the next element)
- the No branch loops back to the input

3.3

```
Step 1 - open Marks.txt; set PassCount to 0 and Highest to 0
Step 2 - read the next mark from the file
Step 3 - if the mark is 50 or more, add 1 to PassCount
Step 4 - if the mark is greater than Highest, set Highest to the mark
Step 5 - repeat from step 2 until the end of the file
Step 6 - output PassCount and Highest; close the file
```

- initialise counters + open; read in a loop to end of file
- count passes; track highest; output; no pseudocode used / clear steps

3.4

- set the largest-so-far to `Score[1]` and its position to 1
- look at each remaining element from index 2 to 40 in turn
- if an element is greater than the largest-so-far, set the largest-so-far to that element and set the position to that index
- after checking all elements, output the largest-so-far and the position