Solutions

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

2.1

- a **variable** can change while the program runs
- a **constant** is fixed

2.2

- `MOD` is the remainder after division
- $23 = 5 \times 4 + 3$, so **`3`**

2.3

- name: **`DECLARE Name : STRING`**
- tax rate: **`CONSTANT TaxRate ← 0.2`**

2.4

- price: **`REAL`**
- box ticked: **`BOOLEAN`**
- grade letter: **`CHAR`**

2.5

- **WHILE** tests before the body, so the body may run **zero times**
- **REPEAT** tests after the body, so it **always runs at least once**

3.1 C

- a **REPEAT (post-condition)** loop always runs at least once
- A and B can run zero times; D is not a loop

3.2

(a)

- repeating a known number of times is **iteration** (a count-controlled / FOR loop)

(b)

- `Total ← 0`; `FOR I ← 1 TO 10`; input and add; output after

```
total ← 0
FOR I ← 1 TO 10
    INPUT mark
    total ← total + mark
NEXT I
OUTPUT total
```

3.3

- a **function** returns a value to where it was called

- a **procedure** does a task but returns no value

3.4

(a)

- `DECLARE Total : INTEGER`
- `DECLARE Time : INTEGER`
- `DECLARE Count : INTEGER`

(b)

- `Total ← 0` before the loop; `FOR Count ← 1 TO 20`; `INPUT Time`; `Total ← Total + Time`; `Average ← Total / 20`; `OUTPUT Average`

```
Total ← 0
FOR Count ← 1 TO 20
    INPUT Time
    Total ← Total + Time
NEXT Count
Average ← Total / 20
OUTPUT Average
```

(c)

- set `Fastest` to the first time (or a very large value) before the loop
- compare each new time with `Fastest`
- if smaller, replace `Fastest`; output it after the loop

(d)

- `137 DIV 60` = **2** minutes
- `137 MOD 60` = **17** seconds
- displayed as **2 minutes 17 seconds**

(e)

- written once and called three times, so a correction is made in **one place**
- easier to read and test
- a **function returns a value**; a procedure does not