Solutions

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

2.1

- `CONSTANT` with a value, then `DECLARE … : REAL`

```
CONSTANT Pi = 3.14159
DECLARE Radius : REAL
```

2.2

- `23 DIV 4 = 5` (integer quotient)
- `23 MOD 4 = 3` (remainder)

2.3

- multiply first (`4 * 2 = 8`), then add 3 → **11**

2.4

- `LENGTH("Computer") = 8`
- `LEFT("Computer", 3) = "Com"`

3.1

- constant for tax, `INPUT Price`, then compute and output

```
CONSTANT TaxRate = 0.20
DECLARE Price : REAL
DECLARE Total : REAL
INPUT Price
Total ← Price * (1 + TaxRate)
OUTPUT "Total: ", Total
```

3.2

- `7 DIV 2 = 3`
- `7 MOD 2 = 1`
- `(5 > 3) AND (2 > 4) = FALSE`
- `NOT (1 = 1) = FALSE`
- `MID("PROGRAM", 2, 3) = "ROG"`

3.3

- `INPUT`, then `TO_UPPER` and `LENGTH`

```
OUTPUT "Enter a word:"
INPUT Word
OUTPUT TO_UPPER(Word)
OUTPUT "Length: ", LENGTH(Word)
```

3.4

(a) `Mark[1]` — `INTEGER`, e.g. `67`

- `PassRate` — `REAL`, e.g. `73.3`
- `TopName` — `STRING`, e.g. `"Wei"`
- `AllPassed` — `BOOLEAN`, e.g. `FALSE`

(b)

```
Count ← 0
FOR Index ← 1 TO 30
    IF Mark[Index] >= 50
      THEN
        Count ← Count + 1
    ENDIF
NEXT Index
PassRate ← Count / 30 * 100
OUTPUT PassRate
```

- initialise the counter; loop over all 30; correct condition; increment inside the `IF`; percentage calculated after the loop

(c) the counter would hold whatever value happened to be in that memory location, or the program would fail with an "unassigned variable" error

- either way the pass count and the rate would be wrong, and the error might not show every time it runs

(d) an IDE provides a **debugger**: breakpoints stop the program at a chosen line

- single-stepping runs one statement at a time
- a watch window shows the changing value of each variable, so the programmer sees exactly where the value first goes wrong

(e) **alpha** testing is done **inside** the development organisation, on incomplete software, by staff

- **beta** testing releases the near-finished software to selected **external users**, who report faults found in real use