Solutions

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

2.1

- tested before the body → **`WHILE`**
- always at least one pass → **`REPEAT`**
- stops when the condition becomes TRUE → **`REPEAT`** (a `WHILE` loop stops when its condition becomes FALSE)
- pre-condition → **`WHILE`**

2.2

- `WHILE` (pre-condition) may run **zero** times
- `REPEAT … UNTIL` (post-condition) runs **at least once**

2.3

(a) `N` becomes 3, 9, 27; the test `27 < 20` is FALSE, so the loop stops: **27**

(b) `N` becomes 30, 10, -10; `-10 < 0` is TRUE, so the loop stops: **-10**

(c) `10 < 5` is FALSE at the first test, so the body never runs: **10**

2.4

- set `Number` to 3 before the loop
- `WHILE Number <= 15 … ENDWHILE`
- output, then add 3 inside the loop

```
Number ← 3
WHILE Number <= 15
    OUTPUT Number
    Number ← Number + 3
ENDWHILE
```

2.5

- `REPEAT … UNTIL` structure
- `INPUT Age` inside the loop
- the condition `Age >= 0 AND Age <= 120`

```
REPEAT
    INPUT Age
UNTIL Age >= 0 AND Age <= 120
```

2.6

- a **pre-condition** loop
- the condition is tested before any word is processed, so the loop can run zero times when `"END"` is typed first

3.1

- a **post-condition (`REPEAT … UNTIL`)** loop, because the password must be entered **at least once** before it can be tested

```
REPEAT
    INPUT Password
UNTIL Password = "OPEN"
```

3.2

- start `Total` at 0; loop while it is under 100; add each input

```
Total ← 0
WHILE Total < 100
    INPUT n
    Total ← Total + n
ENDWHILE
```

3.3

- the variables declared, and the total and the count set to 0
- the first mark input **before** the loop
- a pre-condition loop: `WHILE Mark <> -1`
- the mark added to the total, and 1 added to the count
- the next mark input as the **last** statement in the loop
- an `IF` after the loop that tests whether the count is 0
- `Total / Count` output in one path, and `"No marks"` in the other

```
DECLARE Mark, Total, Count : INTEGER
Total ← 0
Count ← 0
INPUT Mark
WHILE Mark <> -1
    Total ← Total + Mark
    Count ← Count + 1
    INPUT Mark
ENDWHILE
IF Count = 0 THEN
    OUTPUT "No marks"
ELSE
    OUTPUT Total / Count
ENDIF
```

- a `REPEAT` loop does not fit: its body would run once even for an empty class, and the -1 would be counted as a mark

3.4

- use `REPEAT ... UNTIL` when the body must run **at least once** whatever happens, because the condition cannot be tested until the body has run

- the value the condition depends on does not exist before the first pass
- example: asking the user for a password, or for a value in a valid range – you must ask once before you can check what was entered; a `WHILE` would have to test a variable that has not been given a value yet

3.5

(a) **3, 2, 1**

(b) 0 is output, and `Number` becomes -1; the test `Number = 0` is FALSE, and it stays FALSE as `Number` goes further down, so the loop **never ends**

- the body of a post-condition loop runs once **before** the first test, so the value 0 is passed before it can be noticed

(c) a `WHILE` loop with the condition `Number > 0`

- the same two statements in the body
- closed with `ENDWHILE`, and the input before the loop

```
INPUT Number
WHILE Number > 0
    OUTPUT Number
    Number ← Number - 1
ENDWHILE
```

3.6

(a) a loop that ends when the choice is 4; the choice input inside the loop; `CASE OF … ENDCASE`; choices 1 and 2; choice 3; choice 4 and `OTHERWISE`

```
DECLARE Choice : INTEGER
DECLARE Total, Number : REAL
Total ← 1

REPEAT
    INPUT Choice
    CASE OF Choice
        1 : INPUT Number
            Total ← Total + Number
        2 : INPUT Number
            Total ← Total * Number
        3 : OUTPUT Total
        4 : OUTPUT "Stopped"
        OTHERWISE : OUTPUT "Invalid"
    ENDCASE
UNTIL Choice = 4
```

- *(one `IF Choice = 1 OR Choice = 2` before the `CASE` that inputs `Number` is also correct)*

(b) a **post-condition** loop

- the choice must be input at least once before it can be tested