Solutions

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

2.1

- `LENGTH(Names)` is recomputed on every pass, so compute it once into a local variable

```
DECLARE Len : INTEGER
Len <- LENGTH(Names)
FOR Index <- 1 TO Len
    OUTPUT UCASE(MID(Names, Index, 1))
NEXT Index
```

- the rewritten loop, with the declaration

2.2

- any two: repeating a calculation inside a loop whose answer does not change; carrying on looping after the answer is known; testing a condition that an earlier test has already settled; using a loop or a long `CASE` where arithmetic would do

2.3

- input the password: a **procedure** – it performs an action and returns nothing the caller must have
- check whether it is valid: a **function** – it produces one value (`TRUE`/`FALSE`) that the rest of the algorithm uses in an `IF`
- display the welcome screen: a **procedure** – it does something, and there is no value to return

2.4

- a box represents a **module** (a procedure or function)
- an arrow pointing **up** beside a call line represents data **returned** from the lower module to the one that called it

2.5

- modules that have already been tested individually are combined into a single sub-program, which is then tested as a whole

3.1

(a) the inner `IF` repeats a test the outer `IF` has already made

- `Readings[Index]` is read from the array three times in one pass, when once into a local would do; and `Count` is used without being declared

(b)


```
FUNCTION HowMany(Limit : INTEGER) RETURNS INTEGER
    DECLARE Index, Count, ThisOne : INTEGER
    Count <- 0
    FOR Index <- 1 TO 500
        ThisOne <- Readings[Index]
        IF ThisOne > Limit AND ThisOne <> -1 THEN
            Count <- Count + 1
        ENDIF
    NEXT Index
    RETURN Count
ENDFUNCTION
```

- one test instead of two; the element read once into a local; `Count` declared and the return kept outside the loop

3.2

- find a member: a **function** – it produces one value the rest of the program uses `FUNCTION FindMember(CardNumber : STRING) RETURNS INTEGER`
- work out the fine: a **function** – one value, used in an expression `FUNCTION Fine(DaysLate : INTEGER) RETURNS REAL`
- print a receipt: a **procedure** – it performs an action and returns nothing `PROCEDURE PrintReceipt(MemberID : INTEGER, Amount : REAL)`
- sensible names and types are accepted; a module that both finds **and** prints would lose a mark, because it does two jobs

3.3

(a) everything the module uses arrives through its **interface**, so it can be called on its own with chosen test values

- nothing outside it can change its variables, so the same inputs always give the same result and a fault is traceable to that module

(b) any two: the same module can be reused elsewhere in the program or in another program; several programmers can work on different modules at once; a change is made in one place only; the program is easier to read because each module has one job

3.4

- the inner `IF Found = 0` is only needed because the loop keeps running after the answer is known
- `LENGTH(InString)` is recomputed on every pass

```
FUNCTION FirstSpace(InString : STRING) RETURNS INTEGER
    DECLARE Index, Len : INTEGER
    Len <- LENGTH(InString)
    FOR Index <- 1 TO Len
        IF MID(InString, Index, 1) = " " THEN
            RETURN Index
        ENDIF
    NEXT Index
    RETURN 0
ENDFUNCTION
```

- returning immediately removes the flag and the inner test
- a final `RETURN 0` for a string with no space, so a value comes back in both cases