Solutions

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

2.1

- the count is set first, and the loop comes next: C, B
- inside the loop a number is input and then tested: E, D
- the output comes after the loop: **C, B, E, D, A**

2.2

- a task is written as a few large steps, and each step is broken down into smaller steps
- this is repeated until every step is simple enough to be programmed

2.3

- when each step is so simple that it can be written as about one statement of program code

2.4

(a) add 1 to the count

(b) repeat (the following steps) while the mark is not -1

(c) output the total divided by the count

2.5

- declarations and `Count ← 0`
- a post-condition loop that ends when the word is `"STOP"`
- input the word inside the loop
- `IF LENGTH(Word) > 5` adds 1 to `Count`
- output `Count` after the loop

```
DECLARE Word : STRING
DECLARE Count : INTEGER
Count ← 0
REPEAT
    INPUT Word
    IF LENGTH(Word) > 5 THEN
        Count ← Count + 1
    ENDIF
UNTIL Word = "STOP"
OUTPUT Count
```

2.6 any **four** of:

- set the fare to 3.00
- multiply the distance in km by 1.50
- add this amount to the fare
- check whether the time is from midnight to 6 a.m.

- if it is, multiply the fare by 2

3.1 one mark for each point, in a sensible order:

1. Set the count to 0.
2. Input an age.
3. If the age is 0, go to step 6.
4. If the age is less than 18, add 1 to the count.
5. Go back to step 2.
6. Output the count.

3.2 any **five** of:

- set the total mileage, the number of cars and the number of high-mileage cars to zero
- input the mileage of the first car
- repeat until the mileage input is zero
- add each mileage to the total, and add 1 to the number of cars
- if a mileage is more than 100 000, add 1 to the number of high-mileage cars
- input the next mileage at the end of each pass
- after the loop, output the total divided by the number of cars, and the number of high-mileage cars

3.3

- the variables declared, and the two starting values set
- a first input **before** the loop
- a pre-condition loop that continues while the temperature is not -999
- the `IF` that updates the highest temperature
- 1 added to the number of readings, and the next input, inside the loop
- both outputs after the loop

```
DECLARE Temp : REAL
DECLARE Highest : REAL
DECLARE Readings : INTEGER
Highest ← -100
Readings ← 0
INPUT Temp
WHILE Temp <> -999
    IF Temp > Highest THEN
        Highest ← Temp
    ENDIF
    Readings ← Readings + 1
    INPUT Temp
ENDWHILE
OUTPUT Highest, Readings
```

- the value is tested before it is used (step 2 comes before steps 3 and 4), so this is a `WHILE` loop; a `REPEAT` loop would count the -999 as a reading

3.4 any **five** of, in a sensible order:

- set the goods total to 0
- for each item, multiply the price by the quantity
- add each result to the goods total
- if the goods total is less than 50.00, add 5.00 for delivery
- calculate the tax as 20% of this amount
- add the tax to give the invoice total

3.5

(a) one mark for each step:

1. Set the number of passes to 0 and the highest mark to 0.
2. Input the first mark.
3. Repeat steps 4 to 6 until the mark input is -1.
4. If the mark is 50 or more, add 1 to the number of passes.
5. If the mark is greater than the highest mark, set the highest mark to this mark.
6. Input the next mark.
7. Output the number of passes and the highest mark.

(b) set a total and a count of marks to 0 in step 1

- inside the loop, add each mark to the total and add 1 to the count of marks
- after the loop, output the total divided by the count of marks