A-LEVEL COMPUTER SCIENCE • EXERCISE SHEET

11.2.1 Writing, choosing and tracing loops

Name: _____ Class: _____ Date: _____ **Total: 39 marks**

KEY WORDS count-controlled loop 计数循环 • pre-condition loop 前测循环 • post-condition loop 后测循环 • trace table 跟踪表 • rogue value 终止值

Objective

Build the skills to answer exam questions that **write, choose and trace loops** in pseudocode: **count-controlled loops** 计数循环, **pre-condition loops** 前测循环 and **post-condition loops** 后测循环, with `IF` statements, nested `IF` statements and `CASE` structures inside them. This sheet covers syllabus 11.2.

You must be able to:

- write a count-controlled loop, including one with `STEP`
- write a pre-condition loop and a post-condition loop, and change one kind of loop into another
- use the standard loop patterns: a running total, a count, the largest or smallest value, a validation loop and a **rogue value** 终止值
- use an `IF` statement with an `ELSE` clause, **nested** 嵌套 `IF` statements or a `CASE` structure inside a loop
- write nested loops
- justify why one loop structure is better suited to a problem than the others
- dry-run a loop with a **trace table** 跟踪表

1 Worked examples

Study these first. Each one shows a **method** that a question later on this sheet needs.

Three loops, one task

Each of these outputs the numbers 1 to 5.

```
FOR Count ← 1 TO 5
    OUTPUT Count
NEXT Count
```

```
Count ← 1
WHILE Count <= 5
    OUTPUT Count
    Count ← Count + 1
ENDWHILE
```

```
Count ← 1
REPEAT
    OUTPUT Count
    Count ← Count + 1
UNTIL Count > 5
```

- A `FOR` loop sets, tests and increases its counter for you. The other two loops need all three steps written out: set the counter **before** the loop, change it **inside** the loop, and test it.
- `WHILE` repeats **while** its condition is true; `REPEAT` repeats **until** its condition is true. So the two conditions are opposites: `Count <= 5` and `Count > 5`.
- `STEP` changes the counter by a different amount: `FOR Count ← 10 TO 0 STEP -2` gives 10, 8, 6, 4, 2, 0.

Loop patterns: a total and a count

Input the temperature at noon on each of 30 days. Output how many days were below zero and the mean temperature.

```
Total ← 0
BelowZero ← 0
FOR Day ← 1 TO 30
    INPUT Temp
    Total ← Total + Temp
    IF Temp < 0 THEN
        BelowZero ← BelowZero + 1
    ENDIF
NEXT Day
OUTPUT BelowZero
OUTPUT Total / 30
```

- **Running total:** set it to 0 before the loop, and add to it inside the loop.
- **Count:** set it to 0, and add 1 inside an `IF`.
- Declare `Day` and `BelowZero` as `INTEGER`, and `Temp` and `Total` as `REAL`.

Loop pattern: the largest value

Output the highest of the 30 temperatures.

```
FOR Day ← 1 TO 30
    INPUT Temp
    IF Day = 1 THEN
        Highest ← Temp
    ELSE
        IF Temp > Highest THEN
            Highest ← Temp
        ENDIF
    ENDIF
NEXT Day
OUTPUT Highest
```

- Start with the first value, not with 0, because every temperature could be below zero. The `ELSE` clause holds a **nested** `IF` that only runs from day 2 onwards.

A validation loop

Input a mark, and keep asking until it is from 0 to 100.

```
REPEAT
    OUTPUT "Enter a mark from 0 to 100"
    INPUT Mark
UNTIL Mark >= 0 AND Mark <= 100
```

- The mark must be input **at least once** before it can be checked, so a post-condition loop fits.
- With a pre-condition loop, input the mark once **before** the loop and again as the last statement **inside** it, and loop `WHILE Mark < 0 OR Mark > 100`.

A rogue value

Input prices one at a time until -1 is entered, then output the total price. The value `-1` is a rogue value: it ends the loop, and it must **not** be added to the total.

```
Total ← 0
INPUT Price
WHILE Price <> -1
    Total ← Total + Price
    INPUT Price
ENDWHILE
OUTPUT Total
```

- A pre-condition loop is used, because the very first input could be `-1`, and then nothing should be added.
- Inside the loop, **process the value, then input the next one**, so every value is tested before it is used.

Nested loops

Input `Rows` *and output a triangle of stars: 1 star on the first row, 2 on the second, and so on.*

```
INPUT Rows
FOR Row ← 1 TO Rows
    Line ← ""
    FOR Star ← 1 TO Row
        Line ← Line & "*"
    NEXT Star
    OUTPUT Line
NEXT Row
```

- The **inner** loop runs completely for each pass of the **outer** loop.
- `OUTPUT` starts a new line each time, so the stars for one row are joined into `Line` first.
- When `Rows` is 3, the statement inside the inner loop runs $1 + 2 + 3 = 6$ times.

Choosing and justifying a loop

the problem	best loop	justification
process exactly 30 readings	count-controlled	the number of repetitions is known before the loop starts
read values until a rogue value, which could come first	pre-condition	the condition is tested before the first pass, so the loop can run zero times
input a password until it is correct	post-condition	the body must run at least once, because the value to test is input inside the loop

In an exam answer, name the loop **and** give the reason in terms of the problem.

Dry-running a loop

Trace this algorithm for the inputs 4, 7, 2 and 0.

```
Count ← 0
Big ← 0
INPUT N
WHILE N <> 0
    Count ← Count + 1
    IF N > Big THEN
        Big ← N
    ENDIF
    INPUT N
ENDWHILE
OUTPUT Count, " ", Big
```

N	Count	Big	OUTPUT
	0	0	
4	1	4	
7	2	7	
2	3		
0			3 7

- Write a value only when a variable changes, and move down a row as the algorithm runs.

2 Practice

Now use the methods above.

2.1 Rewrite this count-controlled loop as a pre-condition loop that gives the same output. [3]

```
FOR Number ← 3 TO 15 STEP 3
    OUTPUT Number
NEXT Number
```

2.2 Write pseudocode that inputs a person's age, and repeats the input until the age is from 0 to 120. Use a post-condition loop. [3]

2.3 Write pseudocode that inputs the heights of 25 students, one at a time, and outputs the smallest height. [4]

2.4 A program reads words from the user until the user types `"END"`. The user may type `"END"` as the first word. State the most suitable loop structure, and justify your choice.[2]

2.5 Complete the trace table for this algorithm. [3]

```
Total ← 0
FOR Outer ← 1 TO 3
    FOR Inner ← 1 TO Outer
        Total ← Total + Inner
    NEXT Inner
NEXT Outer
OUTPUT Total
```

Outer	Inner	Total	OUTPUT
		0	
1	1	1	

3 Exam-style questions

3.1 A weather station records the rainfall, in mm, for each of the 30 days of a month. An algorithm will:

- input the rainfall for each day, one value at a time
- count the days with no rain
- find the total rainfall for the month
- find the day with the most rain; if two days have the same rainfall, keep the earlier day
- output the count, the total and the day number.

Write pseudocode for the algorithm. Declare any variables you use. [7]

3.2 A calculator program repeats a menu until the user chooses to stop.

choice	action
1	input a number and add it to `Total`
2	input a number and multiply `Total` by it
3	output `Total`
4	output `"Stopped"`, and the program ends

(a) Write pseudocode for the program, using a `CASE` structure inside a loop. `Total` starts at 1, and any other choice outputs `"Invalid"`. [6]

(b) **Justify** the loop structure you used. [2]

3.3 Study this pseudocode.

```
Count ← 0
Total ← 0
INPUT Value
WHILE Value <> 999
    IF Value > 0 THEN
        Total ← Total + Value
        Count ← Count + 1
    ENDIF
    INPUT Value
ENDWHILE
IF Count > 0 THEN
    OUTPUT Total / Count
ELSE
    OUTPUT "No data"
ENDIF
```

(a) Complete the trace table for the inputs 6, -2, 9, 0 and 999. [4]

Value	Count	Total	OUTPUT
	0	0	

(b) State the purpose of the algorithm. [2]

(c) The algorithm is rewritten with a post-condition loop, and `INPUT Value` becomes the first statement inside the loop. **Explain** what goes wrong if the first value input is 999, and how to correct it. [3]

Solutions

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

2.1

- 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
```

2.2

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

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

2.3

- input the first height and store it as the smallest
- a loop that runs for the other 24 heights
- compare each height with the smallest, and replace it if smaller
- output the smallest after the loop

```
DECLARE Count : INTEGER
DECLARE Height, Smallest : REAL
INPUT Height
Smallest ← Height
FOR Count ← 2 TO 25
    INPUT Height
    IF Height < Smallest THEN
        Smallest ← Height
    ENDIF
NEXT Count
OUTPUT Smallest
```

2.4

- 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

2.5

Outer	Inner	Total	OUTPUT
		0	
1	1	1	
2	1	2	
	2	4	
3	1	5	
	2	7	
	3	10	
			10

- `Outer` and `Inner` columns correct; `Total` column correct; output **10**

3.1

- declarations; a count-controlled loop from 1 to 30; input inside the loop
- count the days with rainfall 0; running total
- keep the largest rainfall and its day number, replacing only when greater
- output all three after the loop

```
DECLARE Day, DryDays, WettestDay : INTEGER
DECLARE Rain, Total, MostRain : REAL
DryDays ← 0
Total ← 0
MostRain ← -1
WettestDay ← 0
FOR Day ← 1 TO 30
    INPUT Rain
    IF Rain = 0 THEN
        DryDays ← DryDays + 1
    ENDIF
    Total ← Total + Rain
    IF Rain > MostRain THEN
        MostRain ← Rain
        WettestDay ← Day
    ENDIF
NEXT Day
OUTPUT DryDays
OUTPUT Total
OUTPUT WettestDay
```

- `MostRain` starts at -1, below any real rainfall, so day 1 is always stored first; `>` rather than `>=` keeps the earlier day

3.2

(a) a loop that ends when the choice is 4; input the choice inside the loop

- `CASE OF … ENDCASE`; choices 1 and 2; choice 3; choices 4 and `OTHERWISE`

```
DECLARE Choice : INTEGER
DECLARE Total, Number : REAL
Total ← 1
REPEAT
    INPUT Choice
    IF Choice = 1 OR Choice = 2 THEN
        INPUT Number
    ENDIF
    CASE OF Choice
        1 : Total ← Total + Number
        2 : Total ← Total * Number
        3 : OUTPUT Total
        4 : OUTPUT "Stopped"
        OTHERWISE : OUTPUT "Invalid"
    ENDCASE
UNTIL Choice = 4
```

- *(inputting `Number` inside the clauses for 1 and 2 of the `CASE` structure is also correct)*

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

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

3.3

(a)

Value	Count	Total	OUTPUT
	0	0	
6	1	6	
-2			
9	2	15	
0			
999			7.5

- `Value` column; `Count` column; `Total` column; output **7.5**

(b) it outputs the mean (average) of the **positive** values input

- it stops at the rogue value 999, and outputs `"No data"` if there were no positive values

(c) the test comes at the end of the loop, so 999 is processed before the loop stops

- 999 is positive, so it is added to `Total` and counted, and the output is 999 instead of `"No data"`
- correct it by changing the condition to `Value > 0 AND Value <> 999`