A-LEVEL COMPUTER SCIENCE • EXERCISE SHEET

11.2 Constructs

Name: _____ Class: _____ Date: _____ **Total: 23 marks**

Objective

Build the skills to answer exam questions on **constructs** 结构—selection (`IF`, `CASE`) and the three kinds of loop.

You must be able to:

- write `IF`⋯`ELSE`, **nested** `IF`, and `CASE` structures
- write a **count-controlled** (`FOR`), **pre-condition** (`WHILE`) and **post-condition** (`REPEAT`) loop
- **justify** which loop best suits a problem

1 Worked examples

Study these first. Each one shows the method for a question type used later —follow the steps and you can do the Practice and Exam-style questions yourself.

■ Selection: IF and CASE

```
IF Age >= 18 THEN
    OUTPUT "Adult"
ELSE
    OUTPUT "Minor"
ENDIF
```

For one value against **several** options, a `CASE` is cleaner than deep nested `IF`s:

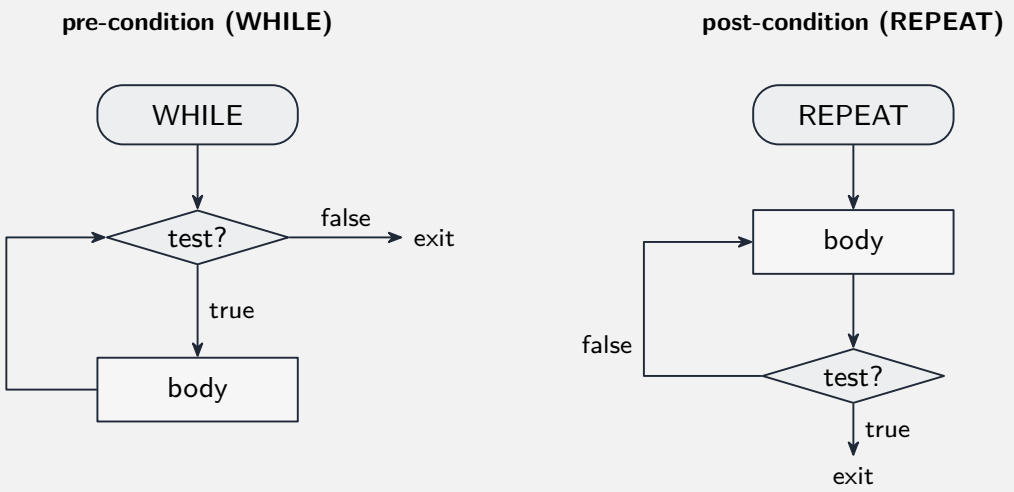
```
CASE OF Grade
    "A": OUTPUT "Excellent"
    "B": OUTPUT "Good"
    OTHERWISE: OUTPUT "Try again"
ENDCASE
```

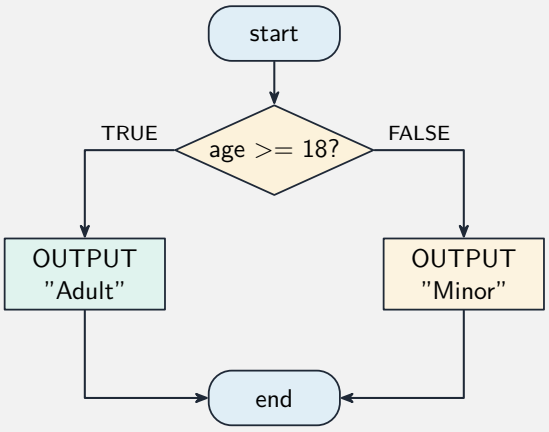
■ The three loops

```
FOR i ← 1 TO 10          // count-controlled — known repeats
    OUTPUT i
NEXT i
```

`WHILE` tests **before** the body (may run **0** times); `REPEAT` tests **after** (runs **at least**

once):





Selection (if-else) is the other core construct

```
WHILE Total < 100 DO          REPEAT
    INPUT n                       INPUT Password
    Total ← Total + n         UNTIL Password = "OPEN"
ENDWHILE
```

Choosing a loop

- repeats **known** in advance → `FOR`
- may need **zero** passes → `WHILE`
- needs **at least one** pass → `REPEAT`

Tracing a loop (trace table)

A **trace table** 追踪表 dry-runs code by hand: one column per variable (plus OUTPUT), one row each time a value changes. Trace this for `N = 3`:

```
Total ← 0
FOR i ← 1 TO N
    Total ← Total + i
NEXT i
OUTPUT Total
```

i	Total	OUTPUT
	0	
1	1	
2	3	
3	6	6

2 Practice

Now apply the methods above.

2.1 Write an `IF…ELSE` that outputs `"Pass"` when `Mark >= 50`, otherwise `"Fail"`. [2]

2.2 Rewrite this nested `IF` as a `CASE` structure. [3]

```
IF Grade = "A" THEN
    OUTPUT "Excellent"
ELSE
    IF Grade = "B" THEN
        OUTPUT "Good"
    ELSE
        OUTPUT "Try again"
    ENDIF
ENDIF
```

2.3 State which loop may run **zero** times and which runs **at least once**. [2]

2.4 Write a `FOR` loop that outputs the **even** numbers from 2 to 20. [2]

3 Exam-style questions

3.1 A menu uses a number `Choice`. Write a `CASE` structure: `1`→`"New"`, `2`→`"Open"`, `3`→`"Save"`, otherwise→`"Invalid"`. [4]

3.2 A program must keep asking for a password until the user types `"OPEN"`. Write the loop using the **most suitable** construct, and **justify** your choice. [3]

3.3 Write a **pre-condition** loop that reads numbers and adds them to a `Total`, stopping once `Total` reaches **100**. [3]

3.4 Complete the **trace table** for this algorithm when the input `X` is `5`. [4]

```
A ← 1
B ← 1
WHILE A < X DO
    B ← B * 2
    A ← A + 1
ENDWHILE
OUTPUT B
```

A	B	OUTPUT
1	1	

4 Go further

You are now ready for the real exam questions on this subtopic. Open the **11.2 Constructs** past-paper sheet in the Library, or try these in **Practice mode**:

- 9618/21 N24 —Q2 (selection and CASE)
- 9618/22 J25 —Q3 (constructs and loops)
- 9618/23 N24 —Q2 (tracing a loop)
- 9618/21 J25 —Q1 (constructs)

Solutions

2.1

```
IF Mark >= 50 THEN
    OUTPUT "Pass"
ELSE
    OUTPUT "Fail"
ENDIF
```

2.2

```
CASE OF Grade
    "A": OUTPUT "Excellent"
    "B": OUTPUT "Good"
    OTHERWISE: OUTPUT "Try again"
ENDCASE
```

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

2.4

```
FOR i ← 2 TO 20 STEP 2
    OUTPUT i
NEXT i
```

(A step of +2 from 2; outputting `i` only when `i MOD 2 = 0` also scores.)

3.1

```
CASE OF Choice
    1: OUTPUT "New"
    2: OUTPUT "Open"
    3: OUTPUT "Save"
    OTHERWISE: OUTPUT "Invalid"
ENDCASE
```

3.2

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

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

3.3

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

3.4 `B` doubles on each pass until `A` reaches 5:

A	B	OUTPUT
1	1	
2	2	
3	4	
4	8	
5	16	16