

9.2 Algorithms

Name: _____ Class: _____ Date: _____ **Total: 40 marks**

KEY WORDS pseudocode 伪代码 • algorithm 算法 • sequence 顺序 • flowchart 流程图
• stepwise refinement 逐步求精

Objective

Build the skills to answer the real exam questions on **algorithms** 算法. In the exam you must **write**, **complete** and **describe** algorithms — not just name the parts. This sheet trains every type the exam uses.

You must be able to:

- complete an **identifier table** 标识符表 for the data a problem uses
- write **pseudocode** 伪代码 using **sequence** 顺序, **selection** 选择 and **iteration** 迭代 (a **loop** 循环), and **declare** 声明 your variables
- write pseudocode that processes an **array** 数组 with a loop
- write pseudocode **from** a **structured English** 结构化英语 description, and **complete** a given flowchart or pseudocode
- use **stepwise refinement** 逐步求精 to break a task into steps
- **describe** an algorithm in words, with no pseudocode

1 Worked examples

Study these first. Each one shows the method for an exam question type used below — read them and you can do the Practice and Exam-style questions yourself.

■ A — Identifier table (the exam format)

Before writing code, name every piece of data. The exam gives you an **example value** and an **explanation**, and you fill in a sensible **variable** 变量 name and **data type** 数据类型:

Example value	Explanation	Variable name	Data type
"Fruit"	a category of stock	Category	STRING
20/02/2025	the date an item was sold	SaleDate	DATE
12.67	the cost of one item	ItemCost	REAL
TRUE	whether the item is in stock	InStock	BOOLEAN

Pick a **descriptive** name (ItemCost, not x). Choose the type from the example: whole number → INTEGER, number with a decimal → REAL, text → STRING, one

true/false → BOOLEAN, a date → DATE.

■ B — The three constructs (quick reference)

```
INPUT Price                                // sequence: steps in order
Total ← Price * Quantity

IF Mark >= 50 THEN                          // selection: choose a path
    OUTPUT "Pass"
ELSE
    OUTPUT "Fail"
ENDIF

FOR Count ← 1 TO 5                          // iteration: a counted loop
    OUTPUT Count
NEXT Count
```

A **WHILE** loop tests **before** each pass (may run 0 times); a **REPEAT ... UNTIL** loop tests **after** (runs at least once). Use ← to assign, = <> < > <= >= to compare.

■ C — Write pseudocode from a structured English description

This is the most common exam task. **Method:**

1. List the data in an identifier table.
2. Take the steps **in order**; turn each into pseudocode.
3. "Repeat ..." → a loop; "if ..." → an IF.

Worked example. *Input 100 numbers one at a time; add up only the positive ones; output the total.*

```
DECLARE Count    : INTEGER
DECLARE Number   : INTEGER
DECLARE Total    : INTEGER
Total ← 0
FOR Count ← 1 TO 100
    INPUT Number
    IF Number > 0 THEN
        Total ← Total + Number
    ENDIF
NEXT Count
OUTPUT Total
```

Total ← 0 **before** the loop, then add inside it — this running-total pattern appears again and again.

■ D — Loop over an array

An array 数组 holds many values under one name. **Data** below has 30 elements; **Data[Index]** is element number **Index**. A FOR loop visits each one.

Worked example. *Output every non-blank element of the array **Data**, then output how many you found.*

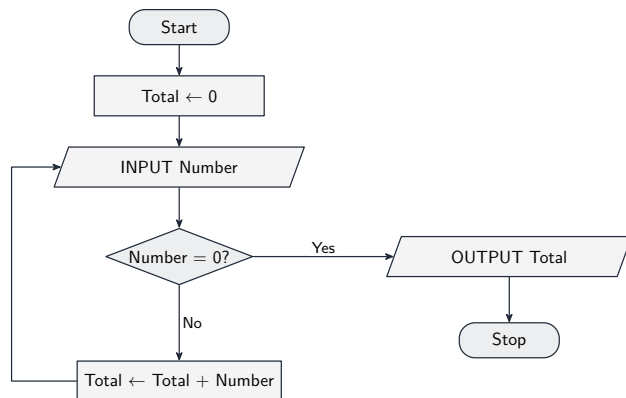
```
DECLARE Index : INTEGER
DECLARE Found : INTEGER
Found ← 0
FOR Index ← 1 TO 30
    IF Data[Index] <> "" THEN
        OUTPUT Data[Index]
        Found ← Found + 1
    ENDIF
NEXT Index
OUTPUT Found
```

(Arrays are covered fully in topic 10 — here you only need **Data[Index]** and a counted loop.)

■ E — Flowcharts: shapes, and a loop

Shape	Meaning
Rounded box	Start / Stop
Parallelogram	Input / Output
Rectangle	Process
Diamond	Decision

A loop is a **decision with an arrow that goes back up**. This flowchart adds input numbers to a total and stops at a **sentinel** 哨兵值 of 0:



■ F — Stepwise refinement

Stepwise refinement 逐步求精 means: start with a few high-level steps, then refine each until it is small enough to code. You write the **steps in words** (no pseudocode).

Worked example. *Average the numbers stored in a file.*

```

Step 1 - open the file; set Total to 0 and Count to 0
Step 2 - read the next number; add it to Total and add 1 to Count
Step 3 - repeat step 2 until the end of the file
Step 4 - set Average to Total / Count
Step 5 - output Average and close the file
  
```

2 Practice

Now apply the methods above.

2.1 Complete the **identifier table** for a program that stores one student's record. [4]

Example value	Explanation	Variable name	Data type
"Ahmed"	the student's name		
16	the student's age in years		
72.5	the student's average mark		
TRUE	whether the student passed		

2.2 Describe what is meant by **stepwise refinement**. [2]

2.3 Write **pseudocode** from this description: *input 20 numbers one at a time; count how many are greater than 100; output the count.* Declare your variables. [5]

2.4 The algorithm in 2.3 uses **sequence**. Name **one** other basic construct it uses and describe how it is used. [2]

2.5 Complete the pseudocode so it averages 10 input values. Write what goes in each blank. [3]

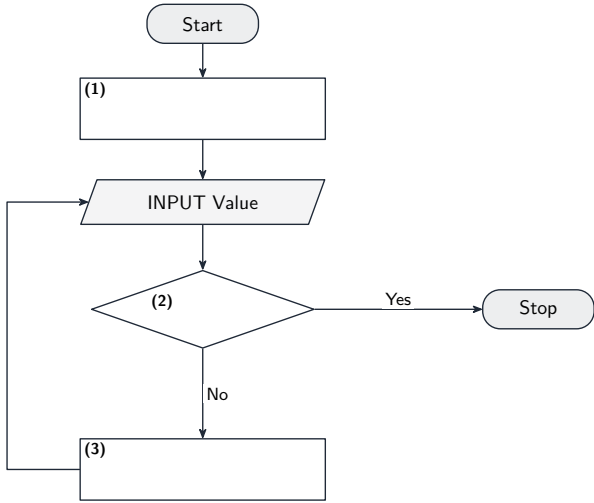
```
Total ← 0
FOR Index ← 1 TO 10
    INPUT Value
    Total ← _____(1)
NEXT Index
Average ← _____(2)
OUTPUT _____(3)
```

3 Exam-style questions

3.1 A program plays a guessing game. A function RANDOM(X, Y) returns a random integer between X and Y inclusive. The program generates a secret integer from 1 to 50, then **repeatedly** inputs a guess until the guess is correct. After each wrong guess it outputs "higher" if the guess was too low, or "lower" if it was too high. When the guess is correct it outputs "correct".

Write **pseudocode** for this algorithm. Declare your variables. [8]

3.2 The array **Data** has 20 elements of type **INTEGER**. The algorithm below inputs values one at a time and stores them in **Data**, stopping when the value -1 is input (the -1 is **not** stored). You may assume at most 20 values are entered. Complete the **three** missing parts of the flowchart. [5]



3.3 A file **Marks.txt** stores exam marks, one per line. Using **stepwise refinement**, describe up to **six** steps for an algorithm that outputs how many marks are a pass (50 or more) and the highest mark. Do **not** use pseudocode. [6]

3.4 An array **Score** holds 40 integers. **Describe** an algorithm (no pseudocode) to find and output the **largest** score and the **position** in the array where it is stored. [5]

Solutions

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

2.1

- Ahmed → Name STRING; 16 → Age INTEGER; 72.5 → Average REAL; TRUE → Passed BOOLEAN

2.2

- start with a few **high-level steps**
- then **break each step down** (refine it) into smaller steps, until every step is simple enough to write as code

2.3

- declare the counters, then loop 20 times with the test inside

```
DECLARE Index  : INTEGER
DECLARE Number : INTEGER
DECLARE Big    : INTEGER
Big ← 0
FOR Index ← 1 TO 20
    INPUT Number
    IF Number > 100 THEN
        Big ← Big + 1
    ENDIF
NEXT Index
OUTPUT Big
```

- IF **Number** > 100 then add 1; OUTPUT after the loop

2.4

- **iteration** — the FOR loop repeats the input-and-test 20 times
- **or selection** — the IF decides whether to add 1

2.5

- (1) Total + Value
- (2) Total / 10
- (3) "Average = ", Average

3.1

- declare and generate **Secret**; loop until the guess matches

```

DECLARE Secret : INTEGER
DECLARE Guess  : INTEGER
Secret ← RANDOM(1, 50)
REPEAT
    INPUT Guess
    IF Guess < Secret THEN
        OUTPUT "higher"
    ENDIF
    IF Guess > Secret THEN
        OUTPUT "lower"
    ENDIF
UNTIL Guess = Secret
OUTPUT "correct"

```

- INPUT inside the loop; too-low → "higher"; too-high → "lower"
- loop ends when `Guess = Secret`; "correct" once after the loop
- (a *WHILE* loop with a first input before it is equally valid)

3.2

- (1) `Count ← 1` (set the array index to the first element)
- (2) decision `Value = -1?` (the sentinel that ends the loop)
- (3) `Data[Count] ← Value` then `Count ← Count + 1` (store, then move to the next element)
- the No branch loops back to the input

3.3

```

Step 1 - open Marks.txt; set PassCount to 0 and Highest to 0
Step 2 - read the next mark from the file
Step 3 - if the mark is 50 or more, add 1 to PassCount
Step 4 - if the mark is greater than Highest, set Highest to the mark
Step 5 - repeat from step 2 until the end of the file
Step 6 - output PassCount and Highest; close the file

```

- initialise counters + open; read in a loop to end of file
- count passes; track highest; output; no pseudocode used / clear steps

3.4

- set the largest-so-far to `Score[1]` and its position to 1
- look at each remaining element from index 2 to 40 in turn
- if an element is greater than the largest-so-far, set the largest-so-far to that element and set the position to that index
- after checking all elements, output the largest-so-far and the position