A-LEVEL COMPUTER SCIENCE • EXERCISE SHEET

9.2.1 Flowcharts, stepwise refinement and describing algorithms

Name: _____ Class: _____ Date: _____ **Total: 43 marks**

KEY WORDS algorithm 算法 • flowchart 流程图 • stepwise refinement 逐步求精 • structured English 结构化英语 • logic statement 逻辑语句

Objective

Build the skills to answer exam questions that **document an algorithm** 算法 in the three ways the exam uses (**structured English** 结构化英语, a program **flowchart** 流程图 and **pseudocode** 伪代码), turn one of them into another, use **stepwise refinement** 逐步求精, write **logic statements** 逻辑语句, and describe an algorithm in words. This sheet covers syllabus 9.2.

You must be able to:

- explain that an algorithm is a solution to a problem expressed as a sequence of defined steps, with input, process and output
- choose identifier names for the data in a problem, and record them in an **identifier table** 标识符表
- document an algorithm as structured English, a flowchart or pseudocode, using sequence, selection and iteration
- draw a flowchart from structured English or from pseudocode, and write pseudocode from structured English or from a flowchart
- use stepwise refinement to break a task down until each step can be programmed
- use logic statements to define parts of an algorithm
- describe an algorithm in words, without pseudocode statements

1 Worked examples

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

■ One algorithm, three ways

A shop till adds up the prices of the items in a basket. The cashier enters 0 after the last item. If the total is more than 100, 10% is taken off. The till outputs the amount to pay.

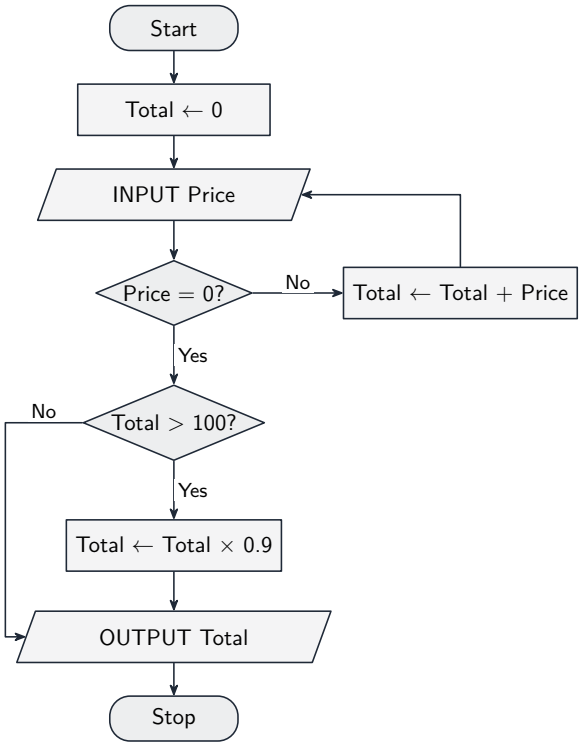
An algorithm is a solution to a problem, written as a **sequence of defined steps**. This one has **input** (the prices), a **process** (adding up and the discount) and **output** (the amount to pay).

identifier	data type	description
`Price`	`REAL`	the price of one item, or 0 to finish
`Total`	`REAL`	the running total, then the amount to pay

Structured English

1. Set the total to 0.
2. Input the price of an item.
3. If the price is 0, go to step 6.
4. Add the price to the total.
5. Go back to step 2.
6. If the total is more than 100, take 10% off the total.
7. Output the total.

Flowchart



Pseudocode

```
DECLARE Price, Total : REAL
Total ← 0
INPUT Price
WHILE Price <> 0
    Total ← Total + Price
    INPUT Price
ENDWHILE
IF Total > 100 THEN
    Total ← Total * 0.9
ENDIF
OUTPUT Total
```

- **Sequence:** the steps run in order. **Selection:** a decision whose two branches join again (the `IF`). **Iteration:** a decision with an arrow that goes back up (the loop).

Turning a flowchart into pseudocode

1. Start at Start and follow the arrows.
2. A rectangle becomes an assignment with ←; a parallelogram becomes `INPUT` or `OUTPUT`.
3. A decision whose two branches **join again** further down becomes `IF … ELSE … ENDIF`.
4. A decision with an arrow that goes **back up** is a loop. If the decision comes **before** the steps it repeats, write a `WHILE` loop; if it comes **after** them, write a `REPEAT … UNTIL` loop.

5. Check the condition: `WHILE` uses the condition for **staying** in the loop, and `UNTIL` uses the condition for **leaving** it.

To draw a flowchart **from** pseudocode or structured English, use the same rules the other way: one shape for each step, and a decision with a labelled **Yes** exit and **No** exit for every `IF`, every loop test and every "go to step".

Stepwise refinement

Stepwise refinement means writing a task as a few large steps, then breaking each step into smaller steps, and repeating this until every step is simple enough to program.

Task: produce a grade report for a class test.

1. Input the marks.
   - 1.1 For each of the 30 students, input the name and the mark.
   - 1.2 If a mark is not from 0 to 50, ask for it again.
2. Work out each grade.
   - 2.1 If the mark is 40 or more, the grade is A.
   - 2.2 Otherwise, if the mark is 25 or more, the grade is B.
   - 2.3 Otherwise, the grade is C.
3. Output the report.
   - 3.1 Output each name with its grade.
   - 3.2 Output how many students got each grade.

- Steps 1 to 3 say **what** must be done. Steps 1.1 to 3.2 say **how**, in enough detail to program each one.
- When a question says "do not use pseudocode", write the steps in plain English, as above.

Logic statements

A **logic statement** is a condition that is TRUE or FALSE. Build it from comparisons joined by `AND`, `OR` and `NOT`.

in words	logic statement
the temperature is from 18 to 24	`Temp >= 18 AND Temp <= 24`
the number is not from 1 to 10	`Number < 1 OR Number > 10`
free delivery for an order over 50, or for a member	`OrderTotal > 50 OR Member = TRUE`
a PIN is valid if it has 4 characters and is not "0000"	`LENGTH(PIN) = 4 AND PIN <> "0000"`

- Every comparison needs both sides: write `Temp >= 18 AND Temp <= 24`, never `Temp >= 18 AND <= 24`.

■ Describing an algorithm in words

When a question says **describe** the algorithm and **do not include pseudocode statements**, give each part as a short numbered sentence. For the till:

1. Set the total to zero.
2. Input the price of the first item.
3. Repeat steps 4 and 5 until the price input is zero.
4. Add the price to the total.
5. Input the price of the next item.
6. If the total is more than 100, reduce it by 10%.
7. Output the total.

- Say which values are set first, what the loop repeats and **when it stops**, each decision, and what is output.
- Words such as `WHILE`, `ENDIF` and ← are pseudocode statements, so leave them out.

2 Practice

Now use the methods above.

2.1 State what is meant by an **algorithm**. [2]

2.2 Write a logic statement for each condition.

(a) `Age` is from 13 to 19. [1]

(b) `UserName` is not an empty string and has at most 12 characters. [1]

(c) A ticket is free when `Age` is under 5 or is 80 or more. [1]

2.3 A gym stores, for each member: the member's name, the number of visits this month, whether this month's fee has been paid, and the monthly fee. Complete the identifier table with a suitable identifier and data type for each item. [4]

identifier	data type	description
		the member's name
		the number of visits this month
		whether this month's fee has been paid
		the monthly fee

2.4 Write pseudocode from this structured English. Declare your variables. [5]

1. Set `Count` to 0.
2. Input a word.
3. If the word has more than 5 characters, add 1 to `Count`.
4. Repeat from step 2 until the word `"STOP"` is input.
5. Output `Count`.

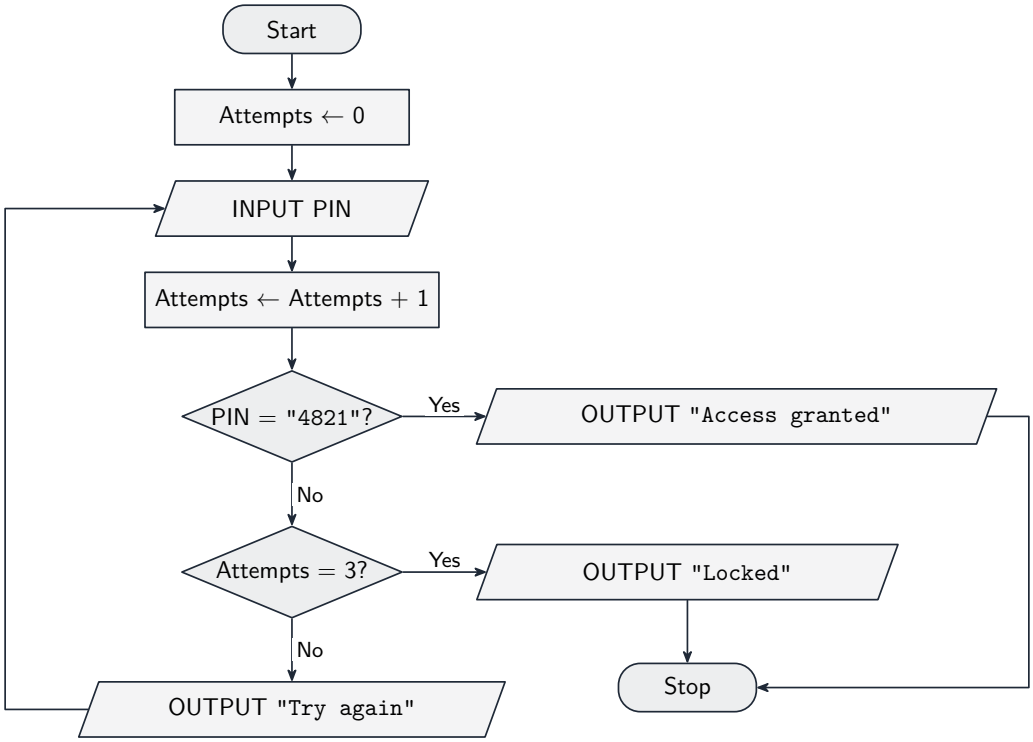
2.5 The task *work out the fare for a taxi journey* has three steps: input the distance and the time; calculate the fare; output the fare. The fare is 3.00, plus 1.50 for each km, and the whole fare is doubled from midnight to 6 a.m. Use stepwise refinement to break the step **calculate the fare** into smaller steps. Do not use pseudocode. [4]

3 Exam-style questions

3.1 A garage records the mileage of each car it services in a day. The number of cars is not known in advance, and 0 is entered after the last car. The algorithm must output the mean mileage and the number of cars with a mileage of more than 100 000.

Describe the algorithm. Do **not** include pseudocode statements in your answer. [5]

3.2 The flowchart shows an algorithm that checks a PIN.



(a) Write pseudocode for the algorithm shown in the flowchart. [6]

(b) State the type of loop you used, and **explain** why it suits this algorithm. [2]

3.3 A ticket machine sells tickets that cost 3.00. It works like this:

1. Set `Paid` to 0.
2. Input the value of a coin.
3. Add the value of the coin to `Paid`.
4. If `Paid` is less than 3.00, go back to step 2.
5. Output the change, which is `Paid` minus 3.00.
6. Output `"Ticket"`.

(a) Draw a program flowchart to represent the algorithm. [5]

(b) State the type of loop in your flowchart, and **explain** your answer. [2]

3.4 A program produces an invoice for a customer's order. At the first level the algorithm has three steps: input the order; calculate the invoice total; output the invoice.

Each item in the order has a price and a quantity. A delivery charge of 5.00 is added if the goods cost less than 50.00, and tax of 20% is then added to the whole amount. Use stepwise refinement to break the step **calculate the invoice total** into at least **four** smaller steps. Do not use pseudocode. [5]

Solutions

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

2.1

- a solution to a problem
- expressed as a sequence of defined steps

2.2

(a) `Age >= 13 AND Age <= 19`

(b) `UserName <> "" AND LENGTH(UserName) <= 12`

(c) `Age < 5 OR Age >= 80`

2.3

- a sensible identifier and type for each row, for example:

identifier	data type	description
`MemberName`	`STRING`	the member's name
`Visits`	`INTEGER`	the number of visits this month
`FeePaid`	`BOOLEAN`	whether this month's fee has been paid
`MonthlyFee`	`REAL`	the monthly fee

2.4

- 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
```

2.5 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 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

3.2

(a) set `Attempts` to 0; a loop containing `INPUT PIN` and adding 1 to `Attempts`

- `"Try again"` only while the PIN is wrong and fewer than 3 attempts have been made
- the loop ends when the PIN is correct **or** 3 attempts have been made
- an `IF` after the loop outputs `"Access granted"` or `"Locked"`

```
DECLARE PIN : STRING
DECLARE Attempts : INTEGER
Attempts ← 0
REPEAT
    INPUT PIN
    Attempts ← Attempts + 1
    IF PIN <> "4821" AND Attempts < 3 THEN
        OUTPUT "Try again"
    ENDIF
UNTIL PIN = "4821" OR Attempts = 3
IF PIN = "4821" THEN
    OUTPUT "Access granted"
ELSE
    OUTPUT "Locked"
ENDIF
```

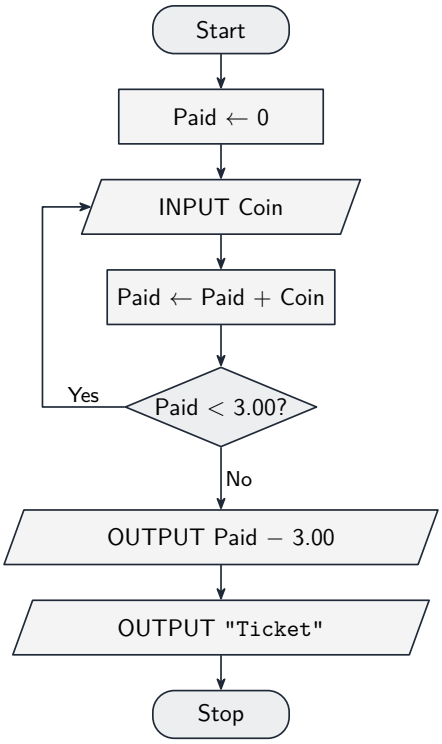
(b) a **post-condition** loop

- the PIN must be input at least once before it can be checked, and the decisions come after the input

3.3

(a) Start and Stop in rounded boxes; `Paid ← 0` in a rectangle

- an input parallelogram, then `Paid ← Paid + Coin`
- a decision `Paid < 3.00?` whose **Yes** exit goes back to the input
- both outputs in parallelograms after the **No** exit



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

- the decision comes **after** the input, so at least one coin is always input

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