

9.2.2 Drawing program flowcharts, and converting between pseudocode and a chart

Name: _____ Class: _____ Date: _____

Total: 31 marks

KEY WORDS algorithm 算法 • pseudocode 伪代码 • flowchart 流程图 • sequence 顺序
• identifier 标识符

Objective

Build the skill that sheets 9.2 and 9.2.1 between them ask for exactly once: **drawing a program flowchart** 流程图. Those sheets write pseudocode, complete an identifier table, use stepwise refinement and **read** a flowchart; the syllabus makes *drawing* one its own statement, in both directions, and a drawing question is usually worth five marks.

You must be able to:

- use the standard flowchart symbols, and use each one for its own purpose only
- draw a flowchart from a **structured English** 结构化英语 description and from **pseudocode** 伪代码
- write pseudocode from a given flowchart
- draw the right loop: a **pre-condition** loop (WHILE) and a **post-condition** loop (REPEAT), and tell them apart by where the decision box sits

1 Worked examples

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

■ The symbols, and the rules that go with them

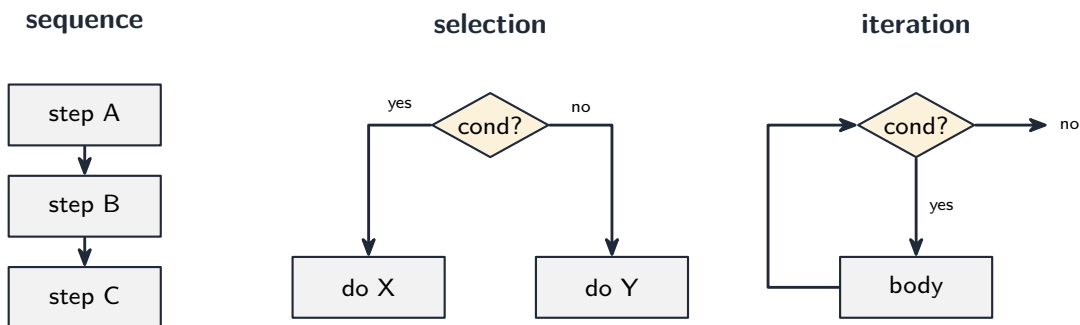
Symbol	Shape	Used for
terminator	rounded box	Start and Stop – one of each, and nothing else
process	rectangle	a calculation or an assignment
input/output	parallelogram	INPUT and OUTPUT only
decision	diamond	a question with a yes/no answer
flow line	arrow	the order the steps run in

Four rules the mark scheme applies:

1. Every symbol except the decision has **one** arrow in and **one** arrow out.

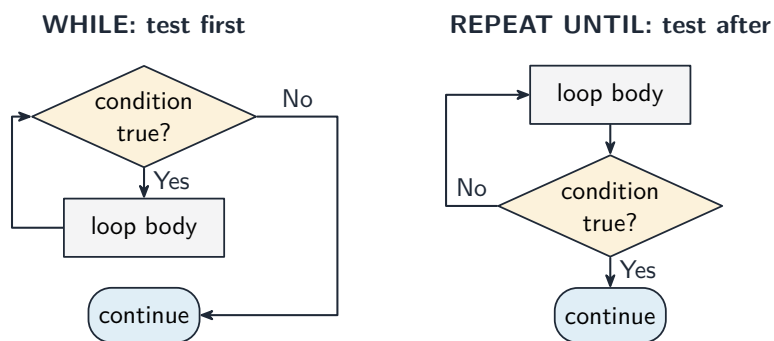
2. A **decision** has exactly **two** exits, and both must be **labelled** – Yes/No or True/False. An unlabelled diamond loses the mark.
3. Every path must end at **Stop**. A path that runs off the page or stops at a box is an error.
4. An assignment is a **process**, not an input. `Total <- 0` goes in a rectangle; only INPUT and OUTPUT go in a parallelogram.

■ The three constructs as shapes



- **Sequence** is boxes stacked in order, one arrow between each pair.
- **Selection** is a diamond with two labelled exits that **rejoin** below. If there is no ELSE, the No branch simply goes straight down to the rejoining point.
- **Iteration** is a diamond whose exit **loops back** upwards to an earlier point.

■ Where the diamond goes tells you which loop it is



This is the single most useful thing to know when converting in either direction.

- **WHILE is a pre-condition loop:** the diamond sits **above** the body, so the body may run **zero** times. Draw the test first, and take the No exit out of the loop.
- **REPEAT ... UNTIL is a post-condition loop:** the body sits **above** the diamond, so the body always runs **at least once**. Draw the body first, and take one exit back up to the top of the body.

So when you are given a flowchart and asked for pseudocode, find the diamond. **Above the body means WHILE; below the body means REPEAT.**

- One more sign-flip to watch: **WHILE** continues while its condition is **true**, but **REPEAT** stops when its condition becomes true. A diamond reading `count < n`? whose **Yes** loops back becomes **UNTIL count >= n** – the **negation**.

■ A worked conversion, pseudocode to flowchart

Draw the flowchart for:

```

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

```

1. **Start** in a rounded box.
2. **INPUT Mark** in a parallelogram.
3. A diamond reading `Mark >= 50?`.
4. From its **Yes** exit, a parallelogram `OUTPUT "Pass"`; from its **No** exit, a parallelogram `OUTPUT "Fail"`.
5. Both arrows **rejoin** and go to **Stop**.

Five boxes, six arrows, two of them labelled. Counting the marks against the symbols like this is what stops a step being dropped.

2 Practice

Now use the methods above.

2.1 Name the flowchart symbol used for each of these.

[4]

- (a) **Start**
- (b) `Total <- Total + Mark`
- (c) **INPUT Name**
- (d) `Is Count = 10?`

2.2 State how many exits a **decision** box has, and state what must be written on them.[2]

2.3 State where the decision box is drawn for a **WHILE** loop and for a **REPEAT** loop, and state what follows from the difference. [3]

2.4 A student's flowchart has a **process** box with two arrows leaving it. State why this is wrong. [1]

3 Exam-style questions

3.1 An algorithm is described in structured English:

1. Input a temperature.
2. If the temperature is above 30, output "Hot"; otherwise output "Not hot".
3. Stop.

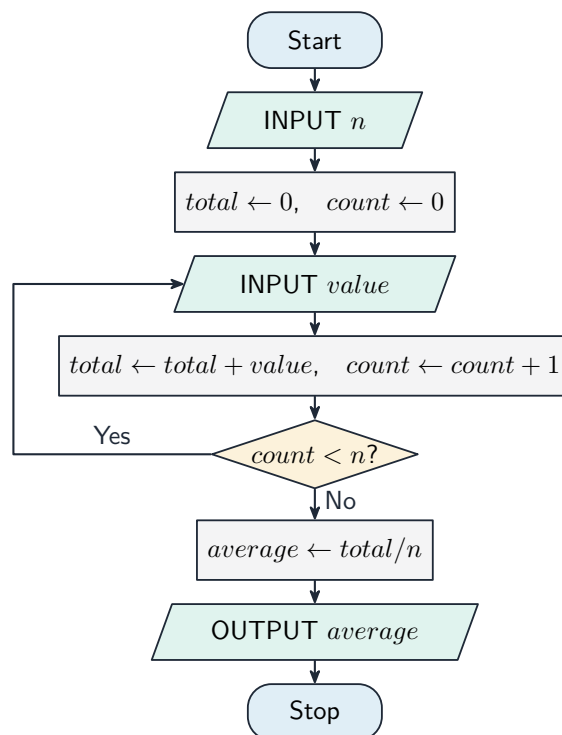
Draw a program flowchart for this algorithm. [5]

3.2 Draw a program flowchart for this pseudocode.

[5]

```
Total <- 0
FOR Count <- 1 TO 5
    INPUT Number
    Total <- Total + Number
NEXT Count
OUTPUT Total
```

3.3 The flowchart shows an algorithm that averages a list of numbers.



Write the pseudocode for this algorithm. Declare your variables.

[5]

3.4 A student draws a flowchart in which:

- the condition is written inside a rectangle,
- the two arrows leaving it carry no labels,
- one of the two paths ends at an output box with no arrow leaving it.

State what is wrong in each case.

[3]

3.5 Explain when an algorithm should use a **REPEAT . . . UNTIL** loop rather than a **WHILE** loop, and give an example of such a task.

[3]
