Exercise walk-through

9.2.2

Drawing program flowcharts, and converting between pseudocode and a chart

A-Level Computer Science

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 The method

Method — 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

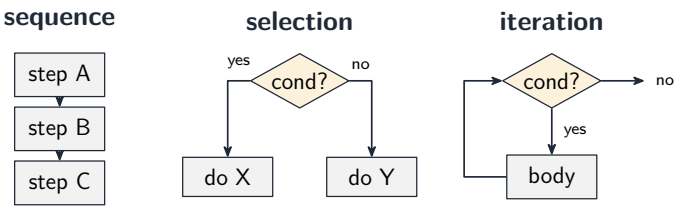
Method — The symbols, and the rules that go with them

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.

Method — 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.



Method — Where the diamond goes tells you which loop it is

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

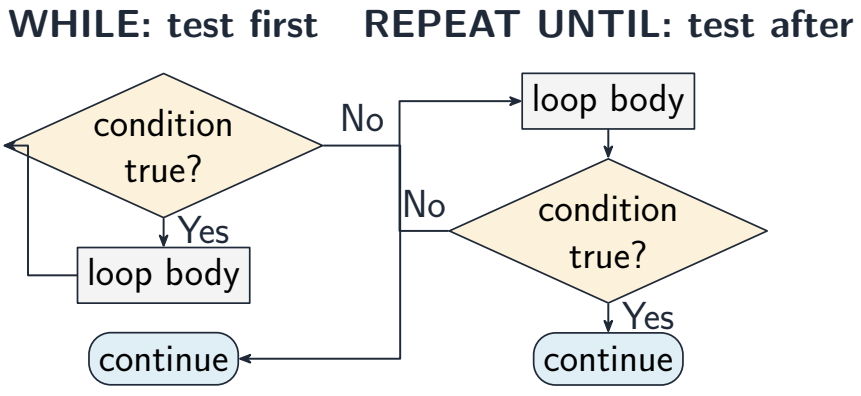
- `WHILE` **is a pre-condition loop**: the diamond sits **above** the body, so the body may run **zero** times. Draw the test first, and take the `No` exit out of the loop.
- `REPEAT ... UNTIL` **is a post-condition loop**: the body sits **above** the diamond, so the body always runs **at least once**.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

Method — Where the diamond goes tells you which loop it is

So when you are given a flowchart and asked for pseudocode, find the diamond.



Method — A worked conversion, pseudocode to flowchart

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

Method — A worked conversion, pseudocode to flowchart

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

Practice 2.1 ▪ 4 marks

Name the flowchart symbol used for each of these.

Practice 2.1 (a)

```
Start
```

Solution 2.1 (a)

Method: The symbols, and the rules that go with them — p.4

Worked steps

a **terminator** – a rounded box B1

Practice 2.1 (b)

```
Total <- Total + Mark
```

Solution 2.1 (b)

Worked steps

a **process** – a rectangle B1

Practice 2.1 (c)

```
INPUT Name
```

Solution 2.1 (c)

Worked steps

an **input/output** symbol – a parallelogram B1

Practice 2.1 (d)

```
Is Count = 10?
```

Solution 2.1 (d)

Worked steps

a **decision** – a diamond B1

Practice 2.2 ▪ 2 marks

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

Solution 2.2

Method: The symbols, and the rules that go with them — p.4

Worked steps

- **two** exits B1
- each must be **labelled**, `Yes` and `No` (or `True` and `False`) B1

Practice 2.3 ▪ 3 marks

State where the decision box is drawn for a `WHILE` loop and for a `REPEAT` loop, and state what follows from the difference.

Solution 2.3

Method: Where the diamond goes tells you which loop it is — p.7

Worked steps

- for a `WHILE` loop the diamond is drawn **above** the body, so the condition is tested **before** the body runs B1
- for a `REPEAT` loop the diamond is drawn **below** the body, so the condition is tested **after** the body runs B1
- so a `WHILE` body may run **zero** times, while a `REPEAT` body always runs **at least once** B1

Practice 2.4 ▪ 1 mark

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

Solution 2.4

Method: A worked conversion, pseudocode to flowchart — p.9

Worked steps

- only a **decision** box may have two exits; a process box does exactly one thing and has one arrow in and one arrow out B1

3 Exam-style

Exam-style 3.1 ▪ 5 marks

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.

Solution 3.1

Method: A worked conversion, pseudocode to flowchart — p.9

Worked steps

- `Start` and `Stop` in rounded boxes **B1**
- `INPUT Temperature` in a parallelogram **B1**
- a diamond reading `Temperature > 30?` **B1**
- `OUTPUT "Hot"` on the `Yes` branch and `OUTPUT "Not hot"` on the `No` branch, both in parallelograms **B1**
- both exits **labelled**, and the two branches rejoining before `Stop` **B1**

Exam-style 3.2 ▪ 5 marks

Draw a program flowchart for this pseudocode.

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

Solution 3.2

Method: A worked conversion, pseudocode to flowchart — p.9

Worked steps

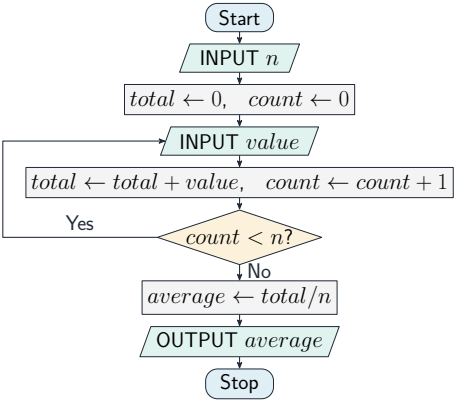
- `Start`, `Stop`, and `Total <- 0` in a process box **B1**
- `Count <- 1` initialised in a process box before the loop **B1**
- `INPUT Number` in a parallelogram and `Total <- Total + Number` in a process box **B1**
- `Count <- Count + 1` and a diamond testing `Count <= 5`?, whose `Yes` exit loops **back** to `INPUT Number` **B1**
- `OUTPUT Total` in a parallelogram on the `No` exit, before `Stop` **B1**
- a `FOR` loop counts a fixed number of times, so drawn as a flowchart it becomes an initialised counter, an increment and a test – the three parts the `FOR` line hides

Exam-style 3.3 ▪ 5 marks

The flowchart shows an algorithm that averages a list of numbers. Write the pseudocode for this algorithm. Declare your variables.

Exam-style 3.3 ▪ 5 marks

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



Solution 3.3

> Method: Where the diamond goes tells you which loop it is — p.7

Worked steps

- the declarations, and `INPUT n` B1
- `total <- 0` and `count <- 0` B1
- a **post-condition** loop, because the diamond is drawn **below** the body B1
- the loop body: `INPUT value`, `total <- total + value`, `count <- count + 1` B1
- `UNTIL count >= n`, then `average <- total / n` and `OUTPUT average` B1

Solution 3.3

```
DECLARE n, count, value, total : INTEGER
DECLARE average : REAL
INPUT n
total <- 0
count <- 0
REPEAT
    INPUT value
    total <- total + value
    count <- count + 1
UNTIL count >= n
average <- total / n
OUTPUT average
```

Solution 3.3

- the diamond reads `count < n?` and its `Yes` loops back, so the `UNTIL` condition is the **negation**, `count >= n`
- a `WHILE` version scores nothing for the loop mark: the chart tests **after** the body, so the body must always run once

Exam-style 3.4 ▪ 3 marks

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.

Solution 3.4

Worked steps

- a condition must be drawn in a **diamond**, not a rectangle: a rectangle is a process and may not have two exits **B1**
- the two arrows leaving a decision must be **labelled** `Yes` and `No`, or the reader cannot tell which path is which **B1**
- every path must reach `Stop`: a box with no arrow leaving it leaves the algorithm unfinished **B1**

Exam-style 3.5 ▪ 3 marks

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

Solution 3.5

> Method: Where the diamond goes tells you which loop it is — p.7

Worked steps

- use `REPEAT ... UNTIL` when the body must run **at least once** whatever happens, because the condition cannot be tested until the body has run B1
- the value the condition depends on does not exist before the first pass B1
- example: asking the user for a password, or for a value in a valid range – you must ask once before you can check what was entered; a `WHILE` would have to test a variable that has not been given a value yet B1