9.2.2 Flowcharts: reading, drawing and converting

Name: _____ Class: _____ Date: _____ **Total: 51 marks**

KEY WORDS flowchart • structured English • pseudocode • pre-condition • post-condition

Objective

Learn to read and to draw a **program flowchart** , and to change an algorithm from one form into another: **structured English** or **pseudocode** into a flowchart, and a flowchart into pseudocode. You need the loops from sheets 11.2.2 and 11.2.3. Sheet 9.2.3 then deals with structured English and stepwise refinement.

You must be able to:

- use the standard flowchart symbols, and use each one for its own purpose only
- follow a flowchart and state what it outputs
- draw a flowchart from a structured English description and from pseudocode
- 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 is

1 Key knowledge and worked examples

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

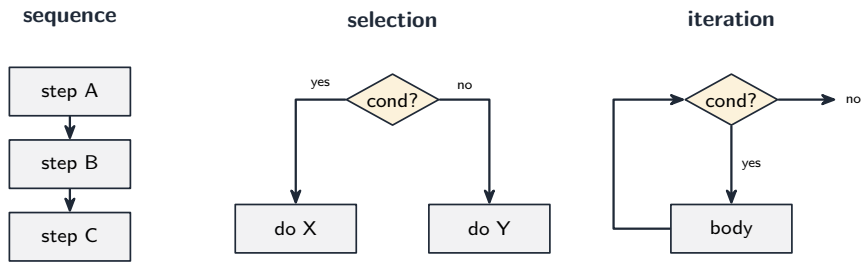
■ The symbols, and the rules that go with them

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:

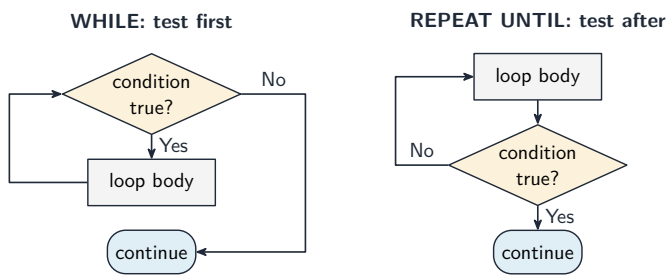
1. Every symbol except the decision has **one** arrow in and **one** arrow out.
2. A **decision** has exactly **two** exits, both **labelled** (`Yes`/`No` or `True`/`False`). An unlabelled diamond loses the mark.
3. Every path ends at `Stop`. A path that runs off the page or stops at a box is an error.
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. With no `ELSE`, the `No` branch 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



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

Flowchart to pseudocode: find the diamond. Above the body → `WHILE`; below the body → `REPEAT`.

- `WHILE` continues while its condition is **true**; `REPEAT` stops when its condition be-

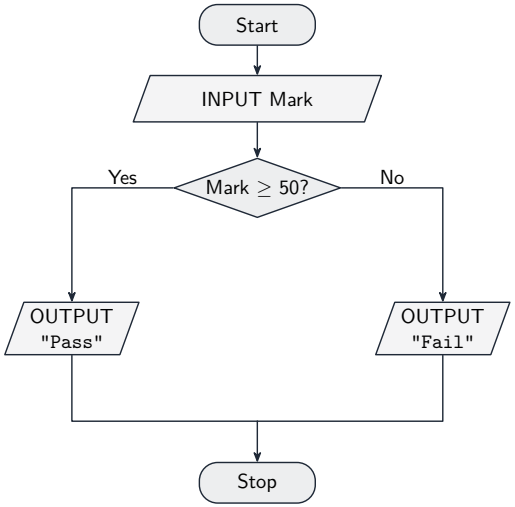
comes **true**. A diamond `count < n?` whose `Yes` loops back becomes `UNTIL count >= n`: the **negation**.

■ 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`.



Six boxes, with both decision exits labelled: count the boxes, so that no step is dropped.

2 Practice

Now use the methods above. The first questions need one idea each.

2.1 Name the flowchart symbol used for each of these. [4]

(a) `Start`

(b) `Total ← Total + Mark`

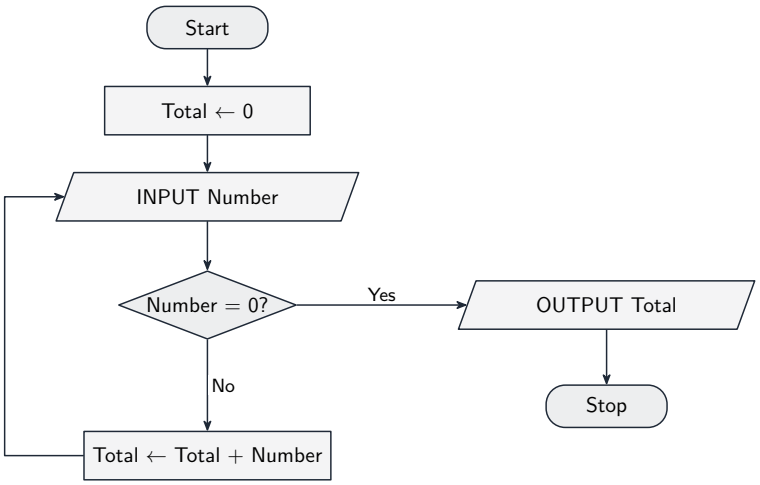
(c) `INPUT Name`

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

2.4 The flowchart adds up numbers that are input.



(a) The numbers 4, 9, 2 and 0 are input, in this order. State the value that is output. [1]

(b) State the input value that makes the algorithm stop. [1]

(c) Name the basic construct that is shown by the arrow that goes back up to the input box. [1]

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

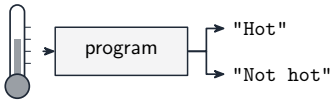
`WHILE` `REPEAT`

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.

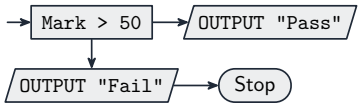
Draw a program flowchart for this algorithm. [5]



3.2 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]



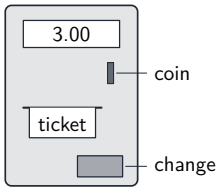
3.3 Draw a program flowchart for this pseudocode. [5]

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

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

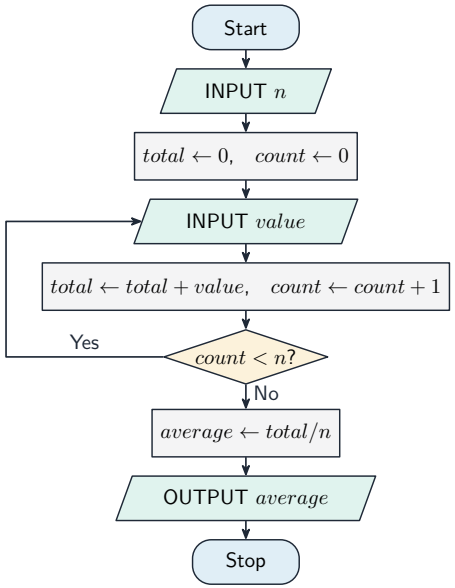
1. Set `Paid` to 0.
2. Input the value of a coin.
3. Add the value of the coin to `Paid`.
4. If `Paid` is less than 3.00, go back to step 2.
5. Output the change, which is `Paid` minus 3.00.
6. Output `"Ticket"`.

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



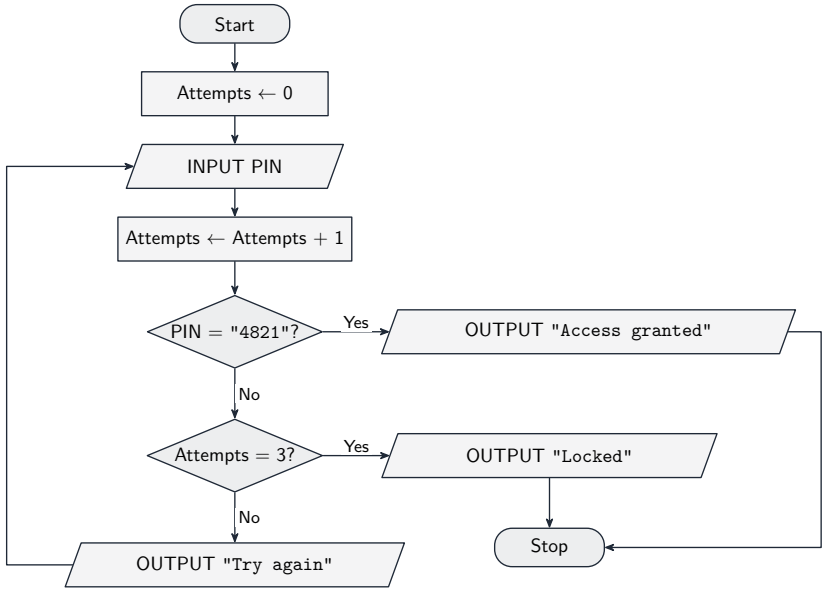
(b) State the type of loop in your flowchart, and **explain** your answer. [2]

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



Write the pseudocode for this algorithm. Declare your variables. [5]

3.6 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]

3.7 Draw a flowchart for an algorithm that repeatedly inputs whole numbers that are zero or greater. The user enters -1 to finish entering numbers. Do not include -1 in the total or the count.

When -1 is entered, output:

- the total of the numbers entered
- how many of those numbers were greater than 10.

Set the total and the count to 0 before the first input. Check whether each input is -1 before adding it to the total. Label both arrows leaving each decision box. [5]

Solutions

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

2.1

(a) a **terminator** – a rounded box

(b) a **process** – a rectangle

(c) an **input/output** symbol – a parallelogram

(d) a **decision** – a diamond

2.2

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

2.3

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

2.4

(a) the loop adds 4, then 9, then 2; the 0 is not added: $4 + 9 + 2 = \mathbf{15}$

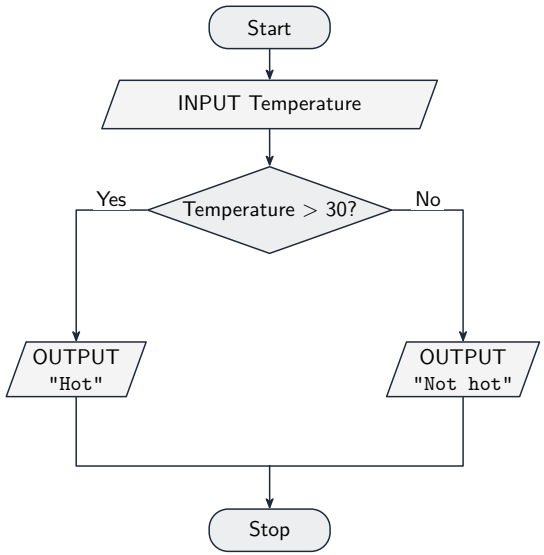
(b) **0**: the decision `Number = 0?` takes the Yes exit to the output

(c) **iteration** (a loop)

2.5

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

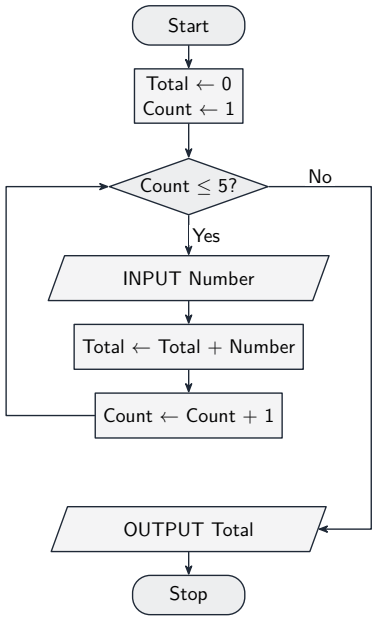
■ 3.1 Flowchart



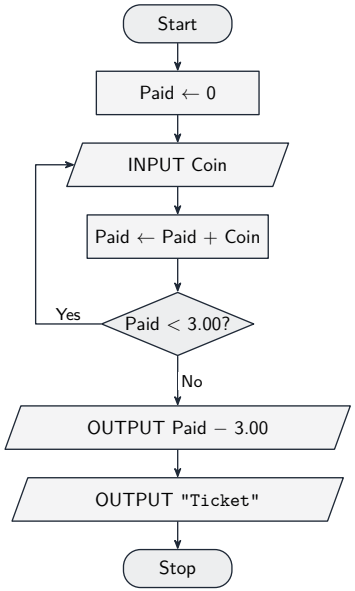
3.2

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

3.3 Flowchart



3.4 (a) Flowchart



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

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

3.5

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

```
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
```

- 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

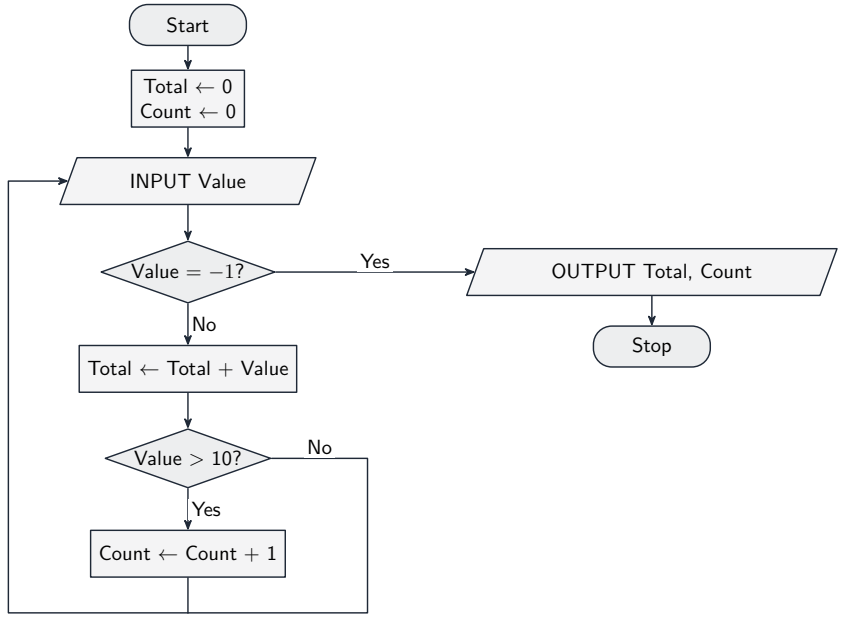
3.6

(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

3.7 Flowchart