

9.2 Algorithms

Name: _____ Class: _____ Date: _____

Total: 62 marks

KEY WORDS algorithm • identifier table • pseudocode • sequence
• data type

Objective

Start designing **algorithms**. On this first sheet of 9.2 you learn what an algorithm is, how to name and list its data in an **identifier table**, and how to write a first algorithm that has **input, process and output**. You need no loops and no IF yet. Sheet 9.2.1 adds decisions, sheet 9.2.2 flowcharts, and sheet 9.2.3 structured English and stepwise refinement.

You must be able to:

- state what an algorithm is
- choose suitable **identifier** names and data types, and complete an identifier table
- say whether a statement is input, process or output
- write **pseudocode** that contains input, process and output, as a **sequence** of steps
- name the three basic constructs: sequence, **selection** and **iteration**

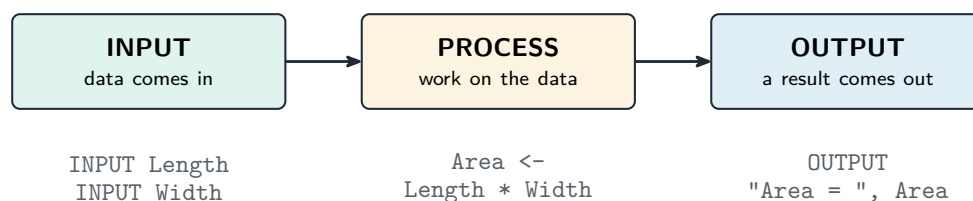
1 Key knowledge and worked examples

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

■ What an algorithm is

An **algorithm** is a **solution to a problem**, expressed as a **sequence of defined steps** (a recipe is one). For a computer each step must be **exact**, and the steps must be in the **right order**.

Almost every algorithm has the same shape:



part	what it does	in pseudocode
input	gets data from outside: keyboard, file, sensor	INPUT RoomLength
process	works on the data: calculates, stores, compares	Area $\leftarrow$ RoomLength * Width
output	gives the result out: screen, file, printer	OUTPUT Area

An **assignment** such as `Total  $\leftarrow$  0` is a **process**: nothing comes in and nothing goes out.

■ Identifiers and the identifier table

Every item of data in an algorithm needs a name, called an **identifier**. Before you write any pseudocode, list the identifiers in an **identifier table**.

list every piece of data first - name, type, description

name	data type	description
ItemCost	REAL	cost of the item
Category	STRING	the product category
SaleDate	DATE	when the item was sold
InStock	BOOLEAN	TRUE if in stock

use **descriptive** names (ItemCost, not x) - the table forces you to plan the data

A good identifier:

- is **meaningful**: it says what the data is (ItemCost, not x or Data1);
- has **no spaces**, starts with a **letter**, and is not a keyword such as INPUT;
- uses capital letters to show the words: ItemCost, NumberOfStudents.

Choose the data type from the values the item can take:

the value is ...	example	data type
a whole number	16	INTEGER
a number with a fraction part	72.5	REAL
text	"Ahmed"	STRING
one character	'A'	CHAR
true or false	TRUE	BOOLEAN
a date	20/02/2025	DATE

A telephone number is a **STRING**: it can start with 0, and nobody adds telephone numbers together.

In the exam the table often gives an example value and an explanation, and you add the name and the type:

Example value	Explanation	Variable name	Data type
"Fruit"	a category of stock	Category	STRING
12.67	the cost of one item	ItemCost	REAL
TRUE	whether the item is in stock	InStock	BOOLEAN

■ Writing a first algorithm: a sequence

Method: (1) list the identifiers; (2) write the inputs; (3) write the process; (4) write the outputs. Declare every variable first with **DECLARE**.

Worked example: input the length and the width of a room in metres, and output its area.

identifier	data type	description
RoomLength	REAL	the length of the room in metres
Width	REAL	the width of the room in metres
Area	REAL	the area of the room in square metres

```

DECLARE RoomLength : REAL
DECLARE Width : REAL
DECLARE Area : REAL
OUTPUT "Enter the length and the width"
INPUT RoomLength
INPUT Width
Area ← RoomLength * Width
OUTPUT "The area is ", Area

```

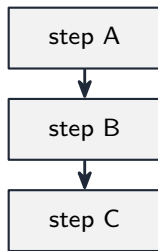
- The arrow $\leftarrow$ means "**becomes**": the value on the right is worked out and stored in the variable on the left.
- The steps run **in order**, one after the other. This is a **sequence**. If $\text{Area} \leftarrow \text{RoomLength} * \text{Width}$ came before the two inputs, the algorithm would not work.

■ The three basic constructs

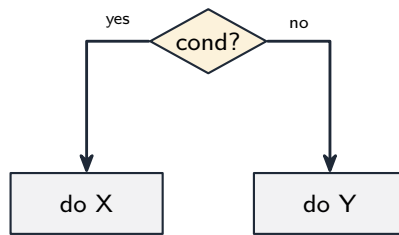
Every algorithm is built from only three **constructs** :

- **sequence**: steps that run one after the other, in order;
- **selection**: a **choice** between paths, which depends on a condition (IF, CASE);
- **iteration**: steps that are **repeated** (FOR, WHILE, REPEAT). Another word is a **loop**.

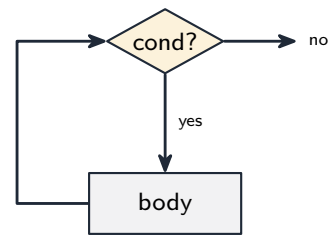
sequence



selection



iteration



On this sheet you only need to **recognise** selection and iteration. You learn to write them on sheets 11.2 to 11.2.3.

■ Key knowledge: evaluating built-in functions

Function	Meaning and example
INT(x)	integer part of a real: $\text{INT}(3.8) \rightarrow 3$
MID(s,start,n)	n characters from position start, counting from 1: $\text{MID}(\text{"ALGORITHM"},3,4) \rightarrow \text{"GORI"}$
LENGTH(s)	number of characters: $\text{LENGTH}(\text{"A B"}) \rightarrow 3$
CHR(n) / ASC(c)	code to character / character to code; use the supplied character-code table
TO_UPPER(s) / TO_LOWER(s)	change letter case: $\text{TO_UPPER}(\text{"As2"}) \rightarrow \text{"AS2"}$
STR_TO_NUM(s) / NUM_TO_STR(x)	numeric text to a number / number to text
&	join strings: $\text{"AS"} \ \& \ \text{"2"} \rightarrow \text{"AS2"}$

Evaluate inner calls first. Check each argument's type; write **ERROR** when the question requests it for an invalid call, such as `LENGTH(27)`. Use `CALL DisplayResult(Total)` for a procedure; a function returns a value used in an expression, such as `Total ← CalculateTotal(Count)`.

Worked example. `TO_UPPER(MID("physics",2,3))` first produces `"hys"`, then `"HYS"`. `STR_TO_NUM("12") + 3` produces the number 15, while `"12" & "3"` produces the string `"123"`.

2 Practice

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

2.1 Give the technical term for a solution to a problem that is expressed as a sequence of defined steps. [1]

1-2-3-

2.2 Tick (✓) **one** box in each row to show whether the statement is input, process or output. [5]

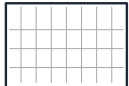


Statement	input	process	output
INPUT Age			
Total $\leftarrow$ Price * Quantity			
OUTPUT "Hello"			
Count $\leftarrow$ 0			
OUTPUT Total			

2.3 A program stores the number of goals a football team has scored. A student suggests these identifiers: `g`, `Goals Scored`, `GoalsScored`, `2ndGoal`.

(a) State which identifier is the most suitable. [1]

(b) Give a reason why **each** of the other three identifiers is not suitable. [3]



2.4 State the most suitable data type for each value.

[5]

- (a) 42
- (b) "Lin Wei"
- (c) 3.75
- (d) FALSE
- (e) 'M'

- (a) 42
- (b) "Lin Wei"
- (c) 3.75
- (d) FALSE
- (e) 'M'

2.5 Complete the identifier table for a program that stores one student's record.

[4]

Ahmed, 16, 72.5, passed

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		

2.6 These pseudocode statements find the total cost of some tickets. They are in the wrong order.

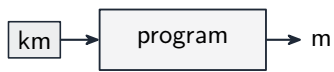
- A OUTPUT TotalCost
- B INPUT NumberOfTickets
- C TotalCost $\leftarrow$ NumberOfTickets * 12.50
- D DECLARE NumberOfTickets : INTEGER and DECLARE TotalCost : REAL

12.50

Write the letters in the correct order.

[2]

2.7 Write pseudocode that inputs a distance in kilometres and outputs the same distance in metres. Declare your variables. (*Hint: copy the shape of the worked example: declare, input, process, output.*) [4]



2.8 Evaluate each expression; write **ERROR** for a type mismatch. Assume character code 65 represents A.

(a) `INT(8.9)` (b) `TO_UPPER(MID("vector",2,3))` (c) `LENGTH("A B")` (d) `CHR(65)` (e) `STR_TO_NUM("15") + 2` (f) `"15" & "2"` (g) `LENGTH(15)` [7]

"vector"

v	e	c	t	o	r
1	2	3	4	5	6

3 Exam-style questions

3.1 A program is being developed for a bicycle-hire shop. An identifier table is used during the design.

(a) Complete the identifier table. [4]

Example value	Explanation	Variable name	Data type
"HR004821"	the code of a hire		
3	the number of hours of the hire		
4.50	the cost of one hour		
TRUE	whether the bicycle has been returned		

(b) An identifier table includes the names of the identifiers. State **two** other pieces of information that an identifier table should contain. [2]



(c) State **two** reasons why an identifier table is written **before** the pseudocode. [2]

3.2 A student's algorithm uses the variables **a**, **b** and **temp2**.

Explain why these names are **not** good practice, and state what the student should do. [2]

a b temp2

3.3 An algorithm is described in structured English:

1. Input the number of hours worked.
2. Input the pay for one hour.
3. Multiply the hours by the pay for one hour to give the pay before tax.
4. Take 20% of this pay as tax.
5. Output the pay after tax.

Write pseudocode for this algorithm. Declare your variables.

[5]



3.4 The table contains extracts of pseudocode. Tick (✓) **one or more** boxes in each row to show which constructs the extract contains.

[4]



Pseudocode extract	assignment	selection	iteration
Total ← Total + 1			
IF Mark > 50 THEN			
FOR Index ← 1 TO 10			
WHILE Count < 5			

3.5 A petrol station wants a program for its pumps. The program inputs the customer's name, the number of litres of fuel taken and the price of one litre. It outputs the cost of the fuel, and then a message that contains the customer's name.

(a) Complete the identifier table for this program. [4]

identifier	data type	description
		the customer's name
		the number of litres of fuel taken
		the price of one litre
		the cost of the fuel

(b) Write pseudocode for the program. Declare your variables. [5]



(c) State which **one** of the three basic constructs your pseudocode uses, and explain how you can tell. [2]

Solutions

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

2.1

- a sequence of defined steps that solves a problem → an **algorithm**

2.2

- INPUT Age brings data in → **input**
- a calculation that is stored → **process**
- OUTPUT "Hello" → **output**
- Count ← 0 is an assignment; nothing comes in or goes out → **process**
- OUTPUT Total → **output**

2.3

(a) **GoalsScored**

(b) g is not meaningful: it does not say what the data is

- Goals Scored contains a space
- 2ndGoal starts with a digit, not a letter

2.4

(a) a whole number → **INTEGER**

(b) text → **STRING**

(c) a number with a fraction part → **REAL**

(d) true or false → **BOOLEAN**

(e) one character → **CHAR**

2.5

- one mark for each row with a meaningful name **and** the correct type, for example:

Example value	Explanation	Variable name	Data type
"Ahmed"	the student's name	StudentName	STRING
16	the student's age in years	Age	INTEGER
72.5	the student's average mark	AverageMark	REAL
TRUE	whether the student passed	Passed	BOOLEAN

2.6

- variables are declared first, and the input must come before the calculation that uses it: D, B
- the cost is worked out before it can be output: D, B, C, A

2.7

- both variables declared with a suitable type
- INPUT of the distance in kilometres
- the distance multiplied by 1000 and stored
- OUTPUT of the result

```
DECLARE Kilometres : REAL
DECLARE Metres : REAL
INPUT Kilometres
Metres ← Kilometres * 1000
OUTPUT Metres
```

2.8

(a) 8; (b) "ECT"; (c) 3; (d) 'A'; (e) 17; (f) "152"; (g) ERROR: LENGTH needs a string. One mark each.

3.1

(a) one mark for each row with a meaningful name **and** the correct type, for example:

Example value	Explanation	Variable name	Data type
"HR004821"	the code of a hire	HireCode	STRING
3	the number of hours of the hire	HoursHired	INTEGER
4.50	the cost of one hour	CostPerHour	REAL
TRUE	whether the bicycle has been returned	Returned	BOOLEAN

(b) the **data type** of each identifier

- a **description** (explanation) of what each identifier is used for

(c) any **two** of:

- the programmer thinks about all the data before writing the logic, so nothing is forgotten
- everybody in the team uses the same name and the same type for the same item
- it shows at once which variables must be declared, and with which types

3.2

- the names are not meaningful: they do not say what data each variable holds, so the algorithm is hard to read, to correct and to change
- the student should use names that describe the data, for example **HoursWorked**

3.3

- all variables declared with suitable types
- two inputs

- $\text{pay before tax} = \text{hours} \times \text{pay for one hour}$
- tax worked out as 20% of the pay, and taken away
- output of the pay after tax

```

DECLARE Hours : REAL
DECLARE PayPerHour : REAL
DECLARE Pay : REAL
DECLARE Tax : REAL
INPUT Hours
INPUT PayPerHour
Pay  $\leftarrow$  Hours * PayPerHour
Tax  $\leftarrow$  Pay * 0.2
Pay  $\leftarrow$  Pay - Tax
OUTPUT Pay

```

3.4

- $\text{Total} \leftarrow \text{Total} + 1$ stores a value $\rightarrow$ **assignment** only
- IF chooses a path $\rightarrow$ **selection** only
- FOR Index $\leftarrow$ 1 TO 10 repeats, and it also gives Index its values $\rightarrow$ **assignment and iteration**
- WHILE repeats $\rightarrow$ **iteration** only

3.5

(a) one mark for each row with a meaningful name **and** the correct type, for example:

identifier	data type	description
CustomerName	STRING	the customer's name
Litres	REAL	the number of litres of fuel taken
PricePerLitre	REAL	the price of one litre
Cost	REAL	the cost of the fuel

(b) the variables declared, with the types from the table

- the three inputs
- the cost worked out as $\text{litres} \times \text{price of one litre}$
- the cost is output
- a message that contains the customer's name is output

```

DECLARE CustomerName : STRING
DECLARE Litres : REAL
DECLARE PricePerLitre : REAL
DECLARE Cost : REAL
INPUT CustomerName
INPUT Litres
INPUT PricePerLitre
Cost  $\leftarrow$  Litres * PricePerLitre
OUTPUT "Cost: ", Cost
OUTPUT "Thank you, ", CustomerName

```

(c) **sequence**

- every statement runs once, in order, from top to bottom: there is no choice between paths and nothing is repeated