Algorithms, flowcharts and pseudocode: AS Computer Science Paper 2 notes

These notes cover the algorithm design and programming content tested in Cambridge International AS and A Level Computer Science (9618) Paper 2, Fundamental Problem-solving and Programming Skills. Paper 2 asks you to read, write, trace and correct algorithms. You must be able to move between three forms of the same algorithm: structured English, a program flowchart and pseudocode. Almost every question uses Cambridge's own pseudocode style, so these notes follow it closely. The exam insert lists the built-in functions you may use, and you should check it in the exam for the exact parameter order.

Work through the sections in order. Learn the boxed definitions word for word, copy the worked examples by hand, and then try the Quick check questions without looking at the answers.

1. Algorithms and the basic constructs

1.1 What an algorithm is

> **Algorithm:** an ordered set of steps to be followed to complete a task or solve a problem.

An algorithm must be precise (each step has one meaning), finite (it must stop) and correct (it must give the right result for every valid input). An algorithm is not the same as a program. The algorithm is the plan; the program is that plan written in a particular programming language so that a computer can run it.

Before writing an algorithm, a programmer uses two thinking skills:

- **Abstraction:** removing unnecessary detail so that only the information needed to solve the problem is kept. For example, a route-finding program needs road lengths and junctions but not the colour of the buildings.
- **Decomposition:** breaking a complex problem into smaller sub-problems that are easier to solve, each of which may become a procedure or function.

Every algorithm, however long, also needs three things: input of data, processing of that data, and output of results. Many algorithms also store data, for example in a file.

1.2 The three basic constructs

Every algorithm can be built from just three control structures.

> **Sequence:** instructions are carried out one after another, in the order in which they are written.
> **Selection:** a condition is tested and the result decides which instructions are carried out next (IF and CASE statements).
> **Iteration:** a group of instructions is repeated, either a set number of times or until a condition is met (FOR, WHILE and REPEAT loops).

Iteration is divided into two kinds:

MEGA LECTURE · Stuck on this topic? Free 1-on-1 demo class WhatsApp a tutor +92 336 7801123
Scan: free demo class

- **Count-controlled iteration:** the number of repetitions is known before the loop starts (FOR loop).
- **Condition-controlled iteration:** the loop repeats until a condition changes, so the number of repetitions is not known in advance. A **pre-condition** loop (WHILE) tests the condition before each repetition, so the body may run zero times. A **post-condition** loop (REPEAT ... UNTIL) tests the condition after each repetition, so the body always runs at least once.

2. Identifier tables

An **identifier** is the name given to a variable, constant, array, procedure or function. Before writing pseudocode, it is good practice to list all the identifiers in an identifier table. Paper 2 questions often give an identifier table or ask you to complete one.

> **Identifier table:** a table listing each identifier used in an algorithm, with its data type and a description of what it stores or does.

Good identifiers are meaningful (TotalCost, not T), use no spaces, start with a letter, and are not reserved words such as INPUT or FOR. Cambridge pseudocode uses mixed case (PascalCase), for example StudentName.

Worked example 1: an identifier table
Problem: input the price of an item and the quantity bought. If the quantity is 10 or more, a discount of 5% applies. Output the total cost.

Identifier	Data type	Description
Price	REAL	price of one item, input by the user
Quantity	INTEGER	number of items bought
DiscountRate	REAL (constant)	fixed discount of 0.05
TotalCost	REAL	cost after any discount, to be output

3. Structured English

Structured English is a way of writing an algorithm using a limited set of English words, written as numbered or indented steps. It is less formal than pseudocode: there are no data types or declarations, but it uses key words such as INPUT, OUTPUT, IF, REPEAT and UNTIL to show the constructs clearly.

> **Structured English:** a method of describing an algorithm using a restricted subset of English words and the basic constructs, written as a series of steps.

Worked example 2: structured English
Problem: a teacher wants to input the marks of 30 students and output how many students passed (50 or more).

MEGA LECTURE · Stuck on this topic? Free 1-on-1 demo class WhatsApp a tutor +92 336 7801123 megalecture.com

1. SET pass count to 0
2. REPEAT 30 times
   2.1 INPUT the next mark
   2.2 IF the mark is 50 or more THEN add 1 to pass count
3. OUTPUT pass count

Structured English is useful at the start of design, when you want to agree what a program should do before thinking about variable names and data types. In Paper 2 you may be asked to turn structured English into pseudocode or a flowchart.

4. Program flowcharts

4.1 Flowchart symbols

A program flowchart shows an algorithm as a diagram. Each step is drawn in a symbol, and arrows (flow lines) show the order in which steps happen. You must know the standard shapes.

Symbol	Shape	Use
Terminator	rounded rectangle (stadium shape)	START and STOP (or the start and end of a procedure)
Input/output	parallelogram	INPUT a value or OUTPUT a result
Process	rectangle	a calculation or assignment, such as Total ← Total + Mark
Decision	diamond	a question with two exits, labelled Yes and No (or True and False)
Predefined process (subroutine)	rectangle with double vertical lines at each side	a call to a procedure or function defined elsewhere
Flow line	arrow	shows the direction of flow from one step to the next

4.2 How constructs appear in a flowchart

- **Sequence:** boxes joined one after another in a straight line.
- **Selection:** a decision diamond with two branches that later join up again.
- **Iteration:** a decision diamond with one branch that leads back up to an earlier step, forming a loop. If the decision is at the top of the loop it is a pre-condition loop (like WHILE); if the decision is at the bottom it is a post-condition loop (like REPEAT).

Worked example 3: describing a flowchart
Problem: input positive numbers until 0 is entered, then output the largest.

1. Terminator: START
2. Process: Largest ← 0
3. Input: INPUT Number
4. Decision: Number = 0? If Yes, go to step 7. If No, go to step 5.

MEGA LECTURE · Stuck on this topic? Free 1-on-1 demo class WhatsApp a tutor +92 336 7801123 megalecture.com

5. Decision: Number > Largest? If Yes, go to step 6. If No, go back to step 3.
6. Process: Largest ← Number, then go back to step 3.
7. Output: OUTPUT Largest
8. Terminator: STOP

The loop is formed by the flow lines from steps 5 and 6 back to step 3. The test in step 4 comes before the processing, so this is a pre-condition loop. The equivalent pseudocode is:

```
Largest ← 0
INPUT Number
WHILE Number <> 0
   IF Number > Largest THEN
      Largest ← Number
   ENDIF
   INPUT Number
ENDWHILE
OUTPUT Largest
```

Notice the "read ahead" pattern: one INPUT before the loop and another as the last statement inside the loop. This makes sure the rogue value 0 is never processed as real data. It is a very common Paper 2 pattern.

Exam tip: when you draw a flowchart, label every exit from a decision box (Yes/No), make every arrow point the right way, and make sure there is exactly one START and at least one STOP. Missing labels and missing arrowheads are the most common reasons for lost marks.

5. Cambridge pseudocode

Pseudocode is a way of writing an algorithm that looks like a programming language but is not tied to any particular language. Cambridge uses a fixed style for 9618. Keywords are written in capital letters and identifiers in mixed case. Indentation shows which statements are inside a selection or loop.

5.1 Data types

Data type	Stores	Example literal
INTEGER	a whole number, positive or negative	42, –7
REAL	a number with a fractional part	3.75, –0.5, 12.0
CHAR	a single character	'A', '9' (single quotes)
STRING	a sequence of characters, possibly empty	"Lahore", "" (double quotes)
BOOLEAN	one of two values	TRUE, FALSE
DATE	a calendar date	normally written dd/mm/yyyy

5.2 Declarations, constants and assignment

MEGA LECTURE · Stuck on this topic? Free 1-on-1 demo class WhatsApp a tutor +92 336 7801123 megalecture.com

Every variable is declared with its data type before it is used:

```
DECLARE StudentName : STRING
DECLARE Age : INTEGER
DECLARE Average : REAL
DECLARE Found : BOOLEAN
```

A constant is a named value that cannot change while the program runs. Using a constant makes a program easier to read and easier to update, because the value only has to be changed in one place.

```
CONSTANT MaxStudents = 30
CONSTANT DiscountRate = 0.05
```

Assignment uses the left arrow ←. The expression on the right is worked out first, and the result is stored in the variable on the left.

```
Count ← 0
Count ← Count + 1
FullName ← FirstName & " " & Surname
```

The & operator joins (concatenates) strings. In Cambridge pseudocode the = sign is used only for comparison, never for assignment.

5.3 Input, output and operators

```
INPUT Mark
OUTPUT "Your mark is ", Mark
```

Type	Operators
Arithmetic	+ – * / (/ always gives a REAL result); also the functions DIV(x, y) for integer division and MOD(x, y) for the remainder
Comparison	= (equal to), <> (not equal to), <, >, <=, >=
Logical	AND, OR, NOT
String	& (concatenation)

For example, DIV(17, 5) = 3 and MOD(17, 5) = 2. MOD is often used to test whether a number is even: MOD(Number, 2) = 0. Comments start with // and are ignored when the algorithm runs.

5.4 Selection: IF statements

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

The ELSE part is optional. IF statements can be nested, but each IF must have its own ENDIF:

```
IF Mark >= 70 THEN
   Grade ← "A"
ELSE
   IF Mark >= 50 THEN
```


MEGA LECTURE · Stuck on this topic? Free 1-on-1 demo class WhatsApp a tutor +92 336 7801123 megalecture.com

```
    Grade ← "B"
  ELSE
    Grade ← "U"
  ENDIF
ENDIF
```

5.5 Selection: CASE statements

When one variable is compared against several values, CASE is clearer than nested IF statements. Each case can be a single value or a range. OTHERWISE catches every value not listed, and must come last.

```
CASE OF Choice
  1 : OUTPUT "New game"
  2 : OUTPUT "Load game"
  3 : OUTPUT "Quit"
  OTHERWISE : OUTPUT "Invalid choice"
ENDCASE
```

```
CASE OF Mark
  70 TO 100 : Grade ← "A"
  50 TO 69 : Grade ← "B"
  OTHERWISE : Grade ← "U"
ENDCASE
```

The cases are checked in order and only the first match is carried out.

5.6 Iteration: FOR loops (count-controlled)

```
FOR Index ← 1 TO 10
  OUTPUT Index * Index
NEXT Index
```

A STEP value changes the increment, and can be negative to count down:

```
FOR Index ← 10 TO 0 STEP -2
  OUTPUT Index
NEXT Index
```

This outputs 10, 8, 6, 4, 2, 0. Never change the loop counter inside a FOR loop.

5.7 Iteration: WHILE loops (pre-condition)

```
INPUT Password
WHILE Password <> "Secret1"
  OUTPUT "Wrong, try again"
  INPUT Password
ENDWHILE
```

The condition is tested before each pass. If it is false at the start, the body is never carried out. The loop continues while the condition is TRUE.

5.8 Iteration: REPEAT loops (post-condition)

```
REPEAT
```


MEGA LECTURE · Stuck on this topic? Free 1-on-1 demo class WhatsApp a tutor +92 336 7801123 megalecture.com

```
    OUTPUT "Enter a mark from 0 to 100"
    INPUT Mark
UNTIL Mark >= 0 AND Mark <= 100
```

The body always runs at least once, and the loop stops when the condition becomes TRUE. This makes REPEAT ideal for input validation.

> **Choosing a loop:**
> FOR: you know how many times to repeat (for example, 30 students, every element of an array).
> WHILE: the body might not need to run at all (for example, reading a file that might be empty, or a rogue-value loop with a read-ahead).
> REPEAT: the body must run at least once (for example, input validation, a menu that is shown until the user chooses Quit).

5.9 String handling

The exam insert gives the string functions. The most used are:

Function	Returns	Example
LENGTH(S)	number of characters in S	LENGTH("Karachi") = 7
LEFT(S, n)	leftmost n characters	LEFT("Karachi", 3) = "Kar"
RIGHT(S, n)	rightmost n characters	RIGHT("Karachi", 2) = "hi"
MID(S, x, n)	n characters starting at position x	MID("Karachi", 3, 3) = "rac"
UCASE(c), LCASE(c)	a character changed to upper or lower case	LCASE('B') = 'b'

Positions in a string start at 1. To process each character, use a FOR loop from 1 to LENGTH(S) and take MID(S, Index, 1). The insert also lists functions such as TO_UPPER, TO_LOWER, NUM_TO_STR, STR_TO_NUM, INT and RAND; check the exact definitions on the insert you are given.

5.10 Arrays

> **Array:** a data structure that holds a fixed number of elements of the same data type under one identifier, where each element is accessed by its index.

A one-dimensional (1D) array is declared with its lower and upper bounds:

```
DECLARE Marks : ARRAY[1:30] OF INTEGER
Marks[1] ← 67
```

A two-dimensional (2D) array has a row index and a column index:

```
DECLARE Seats : ARRAY[1:10, 1:20] OF CHAR
Seats[3, 7] ← 'B'
```

The **lower bound** is the smallest index (here 1) and the **upper bound** is the largest (30 for Marks). Using an index outside the bounds is an error. A 2D array is processed with nested FOR loops: the

MEGA LECTURE · Stuck on this topic? Free 1-on-1 demo class WhatsApp a tutor +92 336 7801123 megalecture.com

outer loop for rows and the inner loop for columns.

Worked example 4: total and average of an array

```
Total ← 0
FOR Index ← 1 TO 30
    Total ← Total + Marks[Index]
NEXT Index
Average ← Total / 30
OUTPUT "Average mark: ", Average
```

5.11 Linear search

A linear search checks each element in turn until the item is found or the end of the array is reached. The array does not need to be sorted.

Worked example 5: linear search with early stop

```
INPUT SearchName
Found ← FALSE
Index ← 1
WHILE Found = FALSE AND Index <= 30
    IF Names[Index] = SearchName THEN
        Found ← TRUE
    ELSE
        Index ← Index + 1
    ENDIF
ENDWHILE
IF Found = TRUE THEN
    OUTPUT SearchName, " found at position ", Index
ELSE
    OUTPUT SearchName, " not found"
ENDIF
```

A WHILE loop is used, not a FOR loop, because the search should stop as soon as the item is found.

5.12 Bubble sort

A bubble sort goes through the array comparing each pair of neighbouring elements and swapping them if they are in the wrong order. After each pass, the largest remaining value has "bubbled" to the end. A flag records whether any swap was made, so the algorithm can stop early when a pass makes no swaps.

Worked example 6: bubble sort (ascending)

```
Top ← 30
REPEAT
    Swapped ← FALSE
    FOR Index ← 1 TO Top - 1
        IF Marks[Index] > Marks[Index + 1] THEN
            Temp ← Marks[Index]
```


MEGA LECTURE · Stuck on this topic? Free 1-on-1 demo class WhatsApp a tutor +92 336 7801123 megalecture.com

```
            Marks[Index] ← Marks[Index + 1]
            Marks[Index + 1] ← Temp
            Swapped ← TRUE
        ENDIF
    NEXT Index
    Top ← Top - 1
UNTIL Swapped = FALSE OR Top = 1
```

The temporary variable Temp is needed because assigning Marks[Index] ← Marks[Index + 1] first would overwrite the original value.

5.13 Procedures and functions

A subroutine is a named block of code that carries out one task and can be called from elsewhere. Using subroutines supports decomposition, avoids repeating code, and lets each part be tested separately.

Procedure: a subroutine that carries out a task and does not return a value; it is called as a statement using CALL.
Function: a subroutine that returns a single value; it is called as part of an expression.
Parameter: a variable named in the subroutine header that receives a value (an argument) when the subroutine is called.

```
PROCEDURE PrintLine(Symbol : CHAR, Size : INTEGER)
    DECLARE Line : STRING
    Line ← ""
    FOR Index ← 1 TO Size
        Line ← Line & Symbol
    NEXT Index
    OUTPUT Line
ENDPROCEDURE

CALL PrintLine('*', 10)
```

```
FUNCTION Larger(A : INTEGER, B : INTEGER) RETURNS INTEGER
    IF A > B THEN
        RETURN A
    ELSE
        RETURN B
    ENDIF
ENDFUNCTION

Biggest ← Larger(Num1, Num2)
```

A function header always states RETURNS and the data type returned, and every route through a function must reach a RETURN statement.

5.14 Passing parameters by value and by reference

By value (BYVAL): a copy of the argument's value is passed; changes made inside the

MEGA LECTURE · Stuck on this topic? Free 1-on-1 demo class WhatsApp a tutor +92 336 7801123 megalecture.com

subroutine do not affect the original variable.
By reference (BYREF): the address of the argument is passed; changes made inside the subroutine change the original variable.

In Cambridge pseudocode, parameters are passed by value unless BYREF is written. BYREF is used when a procedure needs to send results back to the caller, for example swapping two values:

```
PROCEDURE Swap(BYREF X : INTEGER, BYREF Y : INTEGER)
   DECLARE Temp : INTEGER
   Temp ← X
   X ← Y
   Y ← Temp
ENDPROCEDURE
```

If X and Y were passed BYVAL instead, only the copies would be swapped and the caller's variables would stay the same.

A **local variable** is declared inside a subroutine and exists only while that subroutine runs (Temp above). A **global variable** is declared in the main program and can be used anywhere. Local variables are preferred because they stop one part of a program accidentally changing data used by another part.

5.15 Text file handling

Data in variables is lost when a program ends. A text file stores data permanently, one line (string) at a time. A file must be opened in a mode before use and closed afterwards.

Statement	Purpose
OPENFILE "Data.txt" FOR READ	open an existing file to read from it
OPENFILE "Data.txt" FOR WRITE	create a new file (or overwrite an existing file) to write to it
OPENFILE "Data.txt" FOR APPEND	open a file to add lines to the end
READFILE "Data.txt", Line	read the next line into the string variable Line
WRITEFILE "Data.txt", Line	write the string Line to the file as a new line
EOF("Data.txt")	function that returns TRUE when there are no more lines to read
CLOSEFILE "Data.txt"	close the file

Worked example 7: counting lines in a file

```
DECLARE Line : STRING
DECLARE LineCount : INTEGER
LineCount ← 0
OPENFILE "Students.txt" FOR READ
WHILE NOT EOF("Students.txt")
   READFILE "Students.txt", Line
   LineCount ← LineCount + 1
ENDWHILE
CLOSEFILE "Students.txt"
```


MEGA LECTURE · Stuck on this topic? Free 1-on-1 demo class WhatsApp a tutor +92 336 7801123 megalecture.com

OUTPUT "Number of students: ", LineCount
A WHILE loop is used because the file could be empty; EOF is tested before every read.

Exam tip: opening a file FOR WRITE wipes its existing contents. If a question says "add" records to an existing file, use APPEND. Always close every file you open.

6. Stepwise refinement

Stepwise refinement: the process of breaking a problem down into smaller and smaller steps, repeating the breakdown on each step, until every step is simple enough to be written directly as a program statement.

Stepwise refinement is decomposition applied to an algorithm. Each level is numbered so the link to the level above is clear (1, then 1.1, 1.2, then 1.2.1 and so on). The final level should translate line by line into pseudocode.

Worked example 8: stepwise refinement
Problem: a shop records the number of items sold on each of the 7 days of a week. Output the total for the week and the best day.

Level 1:
1. Input the sales for 7 days
2. Calculate the total and find the best day
3. Output the results

Level 2:
1.1 FOR each day from 1 to 7: input that day's sales into Sales[Day], validating that it is not negative
2.1 Set Total to 0, BestDay to 1
2.2 FOR each day from 1 to 7: add Sales[Day] to Total; if Sales[Day] > Sales[BestDay] then set BestDay to Day
3.1 OUTPUT Total
3.2 OUTPUT BestDay and Sales[BestDay]

Level 3 (step 1.1 only):
1.1.1 FOR Day ← 1 TO 7
1.1.2 REPEAT INPUT Sales[Day] UNTIL Sales[Day] >= 0
1.1.3 NEXT Day

Each level-1 step could become its own procedure or function. This is how stepwise refinement leads to a modular program.

7. Trace tables

Trace table: a table used to record the value of each variable, and any output, as an algorithm is

MEGA LECTURE · Stuck on this topic? Free 1-on-1 demo class WhatsApp a tutor +92 336 7801123 megalecture.com

followed step by step by hand (a dry run).

Trace tables are used to find the purpose of an algorithm, to check that it gives the expected output, and to find logic errors. There is one column for each variable, one for any condition being tested (optional) and one for output. Write a new value only when a variable changes, and move down a row each time something changes.

Worked example 9: a trace table

```
Total ← 0
Count ← 0
Num ← 1
WHILE Num <= 10
   IF MOD(Num, 3) = 0 THEN
      Total ← Total + Num
      Count ← Count + 1
   ENDIF
   Num ← Num + 2
ENDWHILE
OUTPUT Total, " ", Count
```

Num	MOD(Num, 3) = 0?	Total	Count	Output
1	FALSE	0	0	
3	TRUE	3	1	
5	FALSE			
7	FALSE			
9	TRUE	12	2	
11	(loop ends)			12 2

Purpose: the algorithm adds up and counts the odd numbers from 1 to 10 that are multiples of 3.

Exam tip: follow the code exactly as written, not as you think it was meant to work. Trace questions often contain a deliberate off-by-one error (for example < instead of <=), and the marks are for spotting the effect.

8. Common errors to avoid

- Using = for assignment. Use ←.
- Forgetting to initialise totals and counters before a loop.
- Putting the initialisation inside the loop, so the total is reset every time.
- NEXT with the wrong variable, or missing ENDIF, ENDWHILE or ENDCASE.
- Comparing a STRING with a number, or a CHAR in double quotes.
- Using a FOR loop where the number of repetitions is not known.

MEGA LECTURE · Stuck on this topic? Free 1-on-1 demo class WhatsApp a tutor +92 336 7801123 megalecture.com

- Forgetting the read-ahead in a rogue-value WHILE loop, so the rogue value is processed.
- Not closing files, or opening FOR WRITE when APPEND was needed.
- A function with a route that never reaches RETURN.

Quick check

Write all algorithms in Cambridge pseudocode unless the question asks for another form.

1. (a) State what is meant by an *algorithm*. [1]
   (b) Name the three basic constructs used in algorithms and describe each one. [3]
2. State the most appropriate type of loop (FOR, WHILE or REPEAT) for each task and justify your choice. [6]
   (a) Output the 12 times table from 1 × 12 to 12 × 12.
   (b) Ask the user for an age until they enter a value from 0 to 120.
   (c) Read and output every line of a text file that may be empty.
3. A program inputs the length and width of a rectangular room in metres and the cost of carpet per square metre, then outputs the total cost of carpeting the room. Complete an identifier table for this program. [4]
4. Write structured English for an algorithm that inputs 20 temperatures and outputs the highest temperature and the average temperature. [5]
5. State the shape and purpose of each of these flowchart symbols: terminator, decision, input/output, process. [4]
6. Write pseudocode to declare: (a) a constant called VATRate with the value 0.17; (b) a 1D array called Prices that can store 50 real numbers; (c) a 2D array called Board of 8 rows and 8 columns that stores characters. [3]
7. Rewrite the following nested IF statements as a single CASE statement. [4]

```
IF Score >= 80 THEN
   Award ← "Gold"
ELSE
   IF Score >= 60 THEN
      Award ← "Silver"
   ELSE
      IF Score >= 40 THEN
         Award ← "Bronze"
      ELSE
         Award ← "None"
      ENDIF
   ENDIF
ENDIF
```

   Score is an integer from 0 to 100.
8. Rewrite this FOR loop as (a) a WHILE loop and (b) a REPEAT loop, so that each gives the same output. [5]

```
FOR Count ← 1 TO 5
   OUTPUT Count * 3
NEXT Count
```


MEGA LECTURE · Stuck on this topic? Free 1-on-1 demo class WhatsApp a tutor +92 336 7801123 megalecture.com

9. Complete a trace table for this algorithm and state the output. [4]

```
X ← 25
Y ← 0
REPEAT
   X ← DIV(X, 2)
   Y ← Y + 1
UNTIL X < 2
OUTPUT Y
```

10. The array Data has the values 4, 9, 2, 7, 5 in elements 1 to 5. Complete a trace table for this algorithm, state the output and state the purpose of the algorithm. [5]

```
Max ← Data[1]
Pos ← 1
FOR i ← 2 TO 5
   IF Data[i] > Max THEN
      Max ← Data[i]
      Pos ← i
   ENDIF
NEXT i
OUTPUT Max, " ", Pos
```

11. Write a function CountVowels that takes a string parameter and returns the number of vowels (a, e, i, o, u, in upper or lower case) in the string. [6]

12. (a) Explain the difference between passing a parameter by value and by reference. [2]
(b) A procedure is defined as PROCEDURE AddBonus(BYVAL Mark : INTEGER), and its only statement is Mark ← Mark + 5. A variable Result holds 60 and the statement CALL AddBonus(Result) is executed. State the value of Result afterwards and explain why. [2]
(c) State what change to the header would make the procedure change Result. [1]

13. A text file Names.txt stores one name per line. Write pseudocode to copy every name that begins with the letter A (upper or lower case) into a new file ANames.txt and then output how many names were copied. [7]

14. A program must process the marks of a class of 30 students: input each mark, check each mark is from 0 to 100, calculate the class average and output the number of students above the average. Use stepwise refinement to break this problem down to two levels. [5]

15. The pseudocode below is meant to input 10 numbers and output their average. Identify four errors and write the correction for each. [4]

```
DECLARE Total : INTEGER
DECLARE Num : INTEGER
FOR Count ← 1 TO 10
   INPUT Num
   Total = Total + Num
NEXT Total
Average ← Total / 10
OUTPUT Average
```

Answers

1. (a) An ordered set of steps to be followed to complete a task or solve a problem.

MEGA LECTURE · Stuck on this topic? Free 1-on-1 demo class WhatsApp a tutor +92 336 7801123 megalecture.com

(b) Sequence: instructions are carried out one after another in the order written. Selection: a condition is tested and the result decides which instructions are carried out next (IF, CASE). Iteration: a group of instructions is repeated a set number of times or until a condition is met (FOR, WHILE, REPEAT).

2. (a) FOR, because the number of repetitions (12) is known before the loop starts.
(b) REPEAT, because the user must enter an age at least once, and the input is checked after it is entered.
(c) WHILE, because the condition (NOT EOF) must be tested before the first read; if the file is empty the loop body must not run at all.

3.

Identifier	Data type	Description
Length	REAL	length of the room in metres, input
Width	REAL	width of the room in metres, input
CostPerSqM	REAL	cost of carpet per square metre, input
TotalCost	REAL	Length × Width × CostPerSqM, output

An extra variable Area (REAL, Length × Width) is also acceptable. Integer types for the lengths lose marks because lengths can have fractional parts.

4.
1. SET highest to the first temperature input and SET total to that temperature
2. REPEAT 19 more times
   2.1 INPUT the next temperature
   2.2 ADD the temperature to total
   2.3 IF the temperature is greater than highest THEN SET highest to the temperature
3. CALCULATE average as total divided by 20
4. OUTPUT highest and average

Setting highest to the first value (not to 0) matters because temperatures can be negative.

5. Terminator: rounded rectangle, shows the START or STOP of the algorithm. Decision: diamond, asks a question with two exits labelled Yes and No. Input/output: parallelogram, shows data being input or results being output. Process: rectangle, shows a calculation or assignment.

6.
```
CONSTANT VATRate = 0.17
DECLARE Prices : ARRAY[1:50] OF REAL
DECLARE Board : ARRAY[1:8, 1:8] OF CHAR
```

7.
```
CASE OF Score
   80 TO 100 : Award ← "Gold"
   60 TO 79 : Award ← "Silver"
   40 TO 59 : Award ← "Bronze"
   OTHERWISE : Award ← "None"
ENDCASE
```
Marks for: CASE OF Score and ENDCASE; correct ranges with no gaps or overlaps; correct

MEGA LECTURE · Stuck on this topic? Free 1-on-1 demo class WhatsApp a tutor +92 336 7801123 megalecture.com

assignments; OTHERWISE for the remaining values.

8. (a)

```
Count ← 1
WHILE Count <= 5
   OUTPUT Count * 3
   Count ← Count + 1
ENDWHILE
```

(b)

```
Count ← 1
REPEAT
   OUTPUT Count * 3
   Count ← Count + 1
UNTIL Count > 5
```

Both output 3, 6, 9, 12, 15. In each case the counter must be initialised before the loop and incremented inside it, which the FOR loop did automatically. Note the conditions are opposites: WHILE continues while Count <= 5; REPEAT stops when Count > 5.

9.

X	Y	X < 2?	Output
25	0		
12	1	FALSE	
6	2	FALSE	
3	3	FALSE	
1	4	TRUE	4

Output: 4. DIV(25, 2) = 12 because integer division discards the remainder. The algorithm counts how many times X can be halved (with integer division) before it falls below 2.

10.

i	Data[i] > Max?	Max	Pos	Output
		4	1	
2	TRUE (9 > 4)	9	2	
3	FALSE (2 > 9)			
4	FALSE (7 > 9)			
5	FALSE (5 > 9)			9 2

Output: 9 2. Purpose: finds the largest value in the array and its position (index).

11.

```
FUNCTION CountVowels(Word : STRING) RETURNS INTEGER
   DECLARE Count : INTEGER
   DECLARE Index : INTEGER
   DECLARE Letter : CHAR
```


MEGA LECTURE · Stuck on this topic? Free 1-on-1 demo class WhatsApp a tutor +92 336 7801123 megalecture.com

```
    Count ← 0
    FOR Index ← 1 TO LENGTH(Word)
        Letter ← LCASE(MID(Word, Index, 1))
        IF Letter = 'a' OR Letter = 'e' OR Letter = 'i' OR Letter = 'o' OR Letter = 'u' THEN
            Count ← Count + 1
        ENDIF
    NEXT Index
    RETURN Count
ENDFUNCTION
```

Marks for: correct header with parameter and RETURNS INTEGER; counter initialised to 0; loop from 1 to LENGTH(Word); extracting each character with MID; handling upper case (LCASE or testing both cases); RETURN Count after the loop. A CASE statement with 'a', 'e', 'i', 'o', 'u' is an equally good way to test the letter.

12. (a) By value: a copy of the value is passed, so changes inside the subroutine do not affect the original variable. By reference: the address of the variable is passed, so changes inside the subroutine change the original variable.
(b) Result is still 60. The parameter is passed BYVAL, so the procedure adds 5 to its own local copy, which is discarded when the procedure ends.
(c) Change the header to PROCEDURE AddBonus(BYREF Mark : INTEGER).

13.

```
DECLARE Name : STRING
DECLARE Copied : INTEGER
Copied ← 0
OPENFILE "Names.txt" FOR READ
OPENFILE "ANames.txt" FOR WRITE
WHILE NOT EOF("Names.txt")
    READFILE "Names.txt", Name
    IF LEFT(Name, 1) = "A" OR LEFT(Name, 1) = "a" THEN
        WRITEFILE "ANames.txt", Name
        Copied ← Copied + 1
    ENDIF
ENDWHILE
CLOSEFILE "Names.txt"
CLOSEFILE "ANames.txt"
OUTPUT Copied, " names copied"
```

Marks for: opening Names.txt FOR READ; opening ANames.txt FOR WRITE; a WHILE NOT EOF loop; READFILE inside the loop; checking the first character for both cases; WRITEFILE and incrementing the counter inside the IF; closing both files and outputting the count. MID(Name, 1, 1) instead of LEFT(Name, 1) is equally correct.

14.
Level 1:
1. Input and validate the 30 marks
2. Calculate the class average
3. Count and output the students above the average
Level 2:

MEGA LECTURE · Stuck on this topic? Free 1-on-1 demo class WhatsApp a tutor +92 336 7801123 megalecture.com

1.1 FOR each student from 1 to 30
1.2 REPEAT input the mark UNTIL it is from 0 to 100
1.3 store the mark in the array Marks
2.1 Set Total to 0
2.2 Add each element of Marks to Total
2.3 Average ← Total / 30
3.1 Set AboveCount to 0
3.2 FOR each mark, IF the mark is greater than Average THEN add 1 to AboveCount
3.3 OUTPUT AboveCount
The marks must be stored in an array, because the average is only known after every mark has been input, and the marks must then be read a second time.

15.
1. Total is never initialised: add Total ← 0 before the loop.
2. Total = Total + Num uses = for assignment: it should be Total ← Total + Num.
3. NEXT Total uses the wrong variable: it should be NEXT Count.
4. Average is not declared, and the result of / is real: add DECLARE Average : REAL. (Count not being declared is also an acceptable answer: DECLARE Count : INTEGER.)

MEGA LECTURE · Stuck on this topic? Free 1-on-1 demo class WhatsApp a tutor +92 336 7801123 megalecture.com