AS · Cambridge (CIE) · Computer Science

2 hours 17 questions

Exam Questions

Algorithms

Algorithm basics / Pseudocode / Translation skills / Refinement & logic

1 An algorithm will output the **last** three lines from a text file Result.txt

The lines need to be output in the same order as they appear in the file.

Assume:

- Three variables LineX, LineY and LineZ will store the three lines. These are of type string and all three variables have been initialised to an empty string.
- The file exists and contains **at least** three lines.

The algorithm to output the lines is expressed in eight steps.

Complete the steps.

1. Open the file .....

2. Loop until .....

3. ..... and store in ThisLine

4. Assign LineY to LineX

5. Assign LineZ to LineY

6. Assign ThisLine to LineZ

7. After the loop, ...

8. Output LineX, LineY, LineZ

Explain the purpose of steps 4, 5 and 6 in the algorithm from part **(a)**.

Answer

Mark Scheme and Guidance

Example answer:

So that the last three lines of the file are output in the correct **order**

A mark for mentioning one of:

- (Ensuring) the lines are **output in correct order**
- Ordering the last three lines
- Ordering the lines so they are in the same order as (they occur) in the file

Max 1 marks

(1 mark)

2 A global integer variable Tick is always incremented every millisecond (1000 times per second) regardless of the other programs running.

The value of Tick can be read by any program but the value should not be changed.

Assume that the value of Tick does not overflow.

As an example, the following pseudocode algorithm would output "Goodbye" 40 seconds after outputting "Hello".

```
DECLARE Start : INTEGER

OUTPUT "Hello"
Start Tick

REPEAT
//do nothing
UNTIL Tick = Start + 40000

OUTPUT "Goodbye"
```

A program is needed to help a user to time an event such as boiling an egg.

The time taken for the event is known as the elapsed time.

The program contains a procedure Timer() which will:

- take two integer values representing an elapsed time in minutes and seconds
- use the value of variable Tick to calculate the elapsed time
- output a warning message 30 seconds before the elapsed time is up

- output a final message when the total time has elapsed

For example, to set an alarm for 5 minutes and 45 seconds the program makes the following call:

CALL Timer(5, 45)

When 5 minutes and 15 seconds have elapsed, the program will output:

"30 seconds to go"

When 5 minutes and 45 seconds have elapsed, the program will output:

"The time is up!"

Write pseudocode for the procedure Timer().

Answer

Mark Scheme and Guidance

Example solution:

```
PROCEDURE Timer(Mins, Secs : INTEGER)
DECLARE WarningTick, EndTick : INTEGER

EndTick ← Tick + 1000 * ((Mins * 60) + Secs)
WarningTick ← EndTick - (30 * 1000)

REPEAT
//do nothing
UNTIL Tick = WarningTick
OUTPUT "30 seconds to go"

REPEAT
//do nothing
UNTIL Tick = EndTick

OUTPUT "The time is up!"
```

ENDPROCEDURE

Mark as follows:
MP1 Procedure heading **and** parameters **and** ending
MP2 'Attempt' to calculate 'total time'/EndTick // 'elapsed time' // WarningTick
MP3 Correct calculation of EndTick **and** WarningTick

MP4 (Design mark)

- Two separate loops – checking warning time then the final time, **OR** ...
- Single loop checking the final time with an IF statement to check for warning time, **OR** ...
- Single loop with two IF statements checking the warning time and final time

MP5 Completely correct MP4
MP6 Output both messages (must be meaningful and follow successful MP4)

(6 marks)

3 (a) A shop sells sandwiches and snacks. The owner chooses a 'daily special' sandwich which is displayed on a board outside the shop. Each 'daily special' has two different fillings and is made with one type of bread.

The owner wants a program to randomly choose the 'daily special' sandwich.

The program designer decides to store the possible sandwich fillings in a 1D array of type string.

The array is declared in pseudocode as follows:

DECLARE Filling : ARRAY [1:35] OF STRING

Each element contains the name of one filling.

An example of the first five elements is as follows:

Index	Element value
1	"Cheese"
2	"Onion"
3	"Salmon"
4	"Anchovies"
5	"Peanut Butter"

A second 1D array stores the possible bread used:

DECLARE Bread : ARRAY [1:10] OF STRING

Each element contains the name of one type of bread.

An example of the first three elements is as follows:

Index	Element value
1	"White"
2	"Brown"
3	"Pitta"

Both arrays may contain unused elements. The value of these will be an empty string and they may occur anywhere in each array.

A procedure Special() will output a message giving the 'daily special' sandwich made from **two** randomly selected different fillings and **one** randomly selected bread.

Unused array elements must **not** be used when creating the 'daily special' sandwich.

Using the above examples, the output could be:

"The daily special is Cheese and Onion on Brown bread."

Complete the pseudocode for the procedure Special().

Assume that both arrays are global.

PROCEDURE Special()

Answer

Mark Scheme and Guidance

Example solution:

```
PROCEDURE Special()
DECLARE Index : INTEGER
DECLARE Filling1, Filling2 : STRING

REPEAT
Index ← INT(RAND(35)) + 1
UNTIL Filling[Index] <> ""

Filling1 ← Filling[Index]

REPEAT
Index ← INT(RAND(35)) + 1
UNTIL Filling[Index] <> "" AND Filling1 <> Filling[Index]

Filling2 ← Filling[Index]

REPEAT
Index ← INT(RAND(10) + 1)
UNTIL Bread[Index] <> ""
```

OUTPUT "The daily special is ", Filling1, " and ", __ Filling2, " on ", Bread[Index], " bread."

ENDPROCEDURE

Mark as follows:
MP1 Loop for Filling 1, avoiding unused elements
MP2 Loop for Filling 2 avoiding unused elements
MP3 Check Filling 2 is different from Filling 1 – could correctly compare either the indices or the array contents

MP4 Loop for Bread, avoiding unused elements

MP5 Using RAND(10) / RAND(35)
MP6 Completely correct use of RAND() - including INT()and +1 in all cases

MP7 Correct output - **once only** – following a reasonable attempt at selection of filings and bread

(7 marks)

(b) The owner decides that some combinations of fillings do not go well together. For example, anchovies and peanut butter.

Describe how the design could be changed to prevent certain combinations being selected.

Answer

Mark Scheme and Guidance

Answers include:

MP1 For each filling, create a list of acceptable / incompatible fillings/indexes
MP2 When selecting the second filling, (as well as checking for an unused element) check that the filling / index is / is not on the list

ALTERNATIVE:
MP1 Create a list of 'good' combinations
MP2 Randomly select from this list

(2 marks)

4 An algorithm will:

1. prompt and input a sequence of 100 integer values, one at a time
2. sum the positive integers
3. output the result of the sum.

Write pseudocode for the algorithm.

Assume the value zero is neither positive nor negative.

You must declare all variables used in the algorithm.

Answer

Mark Scheme and Guidance

```
DECLARE Count, Total, NextNumber : INTEGER

Total ← 0
FOR Count ← 1 TO 100
OUTPUT "Input an integer value"
INPUT NextNumber
IF NextNumber > 0 THEN
Total ← Total + NextNumber
END IF
NEXT Count

OUTPUT Total
```

Mark as follows:
MP1 Declarations of all variables used
MP2 Loop for 100 iterations
MP3 Prompt **and** input a value **in a loop and**
MP4 Test for value > 0 // >=1 **in a loop**
MP5 Sum the Total **in a loop**
MP6 Output of Total after the **loop**

Max 5 marks

(5 marks)

5 An examination paper has a maximum of 75 marks. One of five pass grades (A to E) is assigned, depending on the mark obtained. The lowest mark for a given grade is known as the grade boundary.

A program is being written to process examination marks.

The five grade boundaries are stored in a global 1D array GB of type INTEGER, for example:

Index	Value	Comment
1	65	The minimum mark for an A grade.
2	57	The minimum mark for a B grade.
3	43	The minimum mark for a C grade.
4	35	The minimum mark for a D grade.
5	27	The minimum mark for an E grade.

Any paper that achieves a mark within 2 marks of a grade boundary must be checked. Using the given table, a paper with 45 marks would need to be checked.

The pseudocode algorithm to determine whether a paper should be checked is as shown. The mark for the paper is stored in variable Mark. Global variables Mark, Index, Upper and Lower are declared as integers.

Complete the pseudocode.

```
FOR Index ← 1 TO .....

Lower ← GB[Index] - 2
```

Upper ← .....

IF Mark ..... AND Mark ..... THEN

OUTPUT "Check this paper"

ENDIF

NEXT Index

Answer

Mark Scheme and Guidance

One mark per highlighted part:

FOR Index ← 1 TO **5**
Lower ← GB[Index] - 2
Upper ← **GB[Index] + 2** // **Lower + 4**
IF Mark **>= Lower** AND Mark **<= Upper** THEN
//IF Mark **<= Upper** AND Mark **>= Lower** THEN
OUTPUT "Check this paper"
ENDIF
NEXT Index

(4 marks)

6 In some countries, on the third Sunday in March, daylight saving time begins when clocks move forward by one hour.

A module AdjustClock() will take an integer parameter representing a year. The module will return an integer value representing the number of the day in March on which the clocks move forward.

For example, the following line of pseudocode would assign DayNumber the value 20:

DayNumber ← AdjustClock(2022)

Write pseudocode for the function AdjustClock().

Date functions from the **insert** should be used in your solution.

Answer

Mark Scheme and Guidance

Loop Solution
Example solution:

```
FUNCTION AdjustClock(ThisYear : INTEGER) RETURNS INTEGER
DECLARE ThisDayNumber, SundayCount : INTEGER
DECLARE ThisDate : DATE

ThisDayNumber ← 0
SundayCount ← 0
REPEAT
ThisDayNumber ← ThisDayNumber + 1
ThisDate ← SETDATE(ThisDayNumber, 3, ThisYear)
IF DAYINDEX(ThisDate) = 1 THEN
SundayCount ← SundayCount + 1
ENDIF
UNTIL SundayCount = 3
RETURN ThisDayNumber
ENDFUNCTION
```

Mark as follows:
MP1 Function heading, parameter, ending and return type
MP2 Declare local integer variable that is used to create a date
MP3 Loop until 3rd Sunday found
MP4 Attempt to use both SETDATE() **and** DAYINDEX() **in a loop**
MP5 Correctly generate value of type DATE using SETDATE() **in a loop**
MP6 Test if value represents a Sunday using DAYINDEX() **in a loop**
MP7 Increment Sunday count **in a loop and** initialised correctly before loop
MP8 Return day number of **third** Sunday

Max 7 marks

Non-loop solution
Example solution:

```
FUNCTION AdjustClock(ThisYear : INTEGER) RETURNS INTEGER
DECLARE FirstDayIndex: INTEGER
DECLARE ThisDate : DATE

ThisDate ← SETDATE(1, 3, ThisYear)
FirstDayIndex ← DAYINDEX(ThisDate)
IF FirstDayIndex = 1 THEN
ThirdSunday ← FirstDayIndex + 14 //First day is Sunday
ELSE
ThirdSunday ← 23 – FirstDayIndex // Other days
ENDIF
RETURN ThirdSunday
ENDFUNCTION
```

Mark as follows:
MP1 Function heading, parameter, ending and return type
MP2 Declare local integer variable that is used to hold a day number
MP4 Attempt to use both SETDATE() **and** DAYINDEX()
MP5 Correctly generate value of type DATE using SETDATE() for first of March / specific day in March
MP6 Test if date represents a Sunday / specific day using DAYINDEX() and calculate third Sunday // Correctly calculate third Sunday for a value returned by DAYINDEX()for one day
MP7 Calculate third Sunday for other six days
MP8 Return day number of **third** Sunday

Max 7 marks

(7 marks)

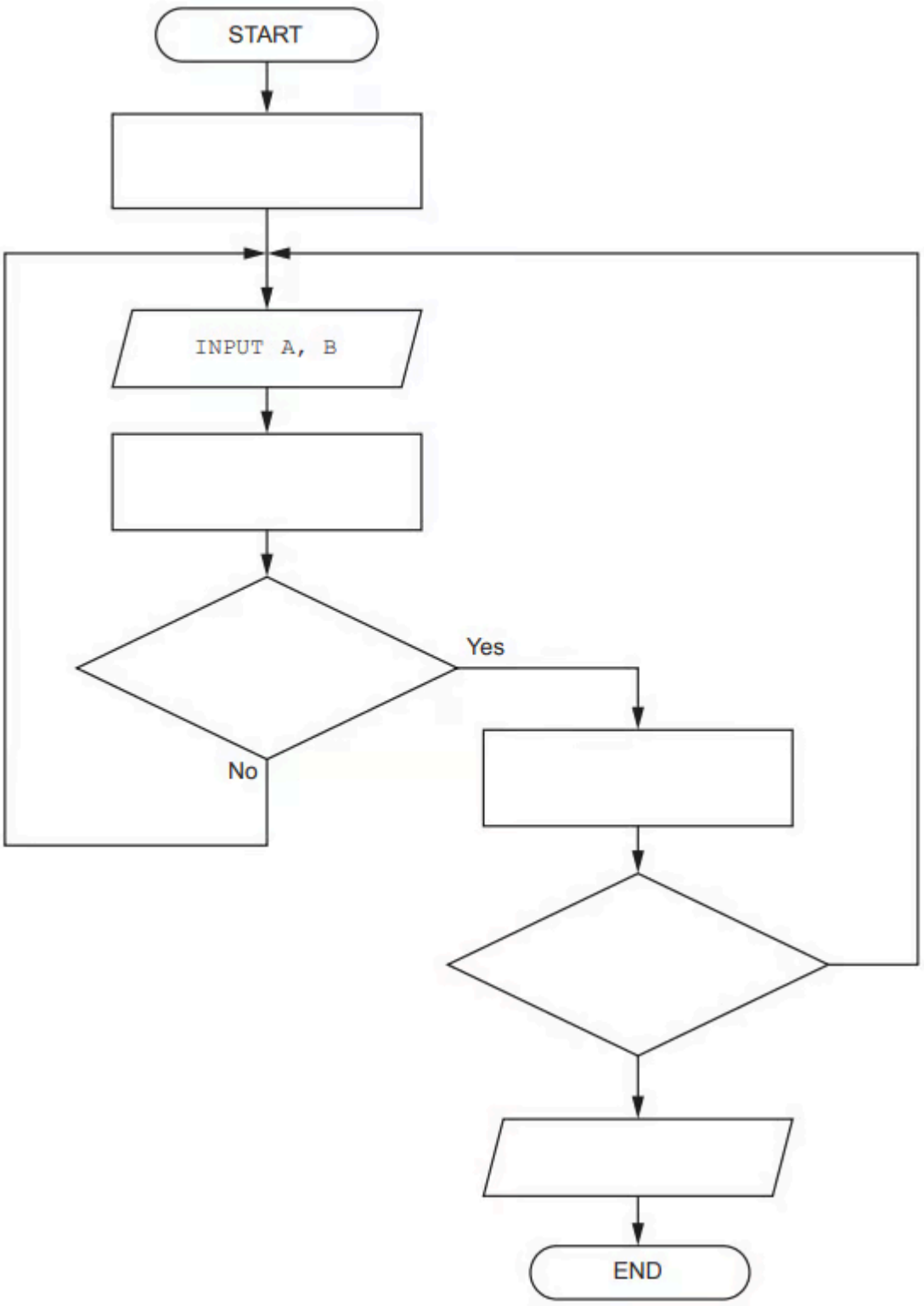
7 An algorithm has three steps. It will:

1. repeatedly input a pair of numeric values A and B

2. count the number of pairs that are input until A has been greater than B 10 times

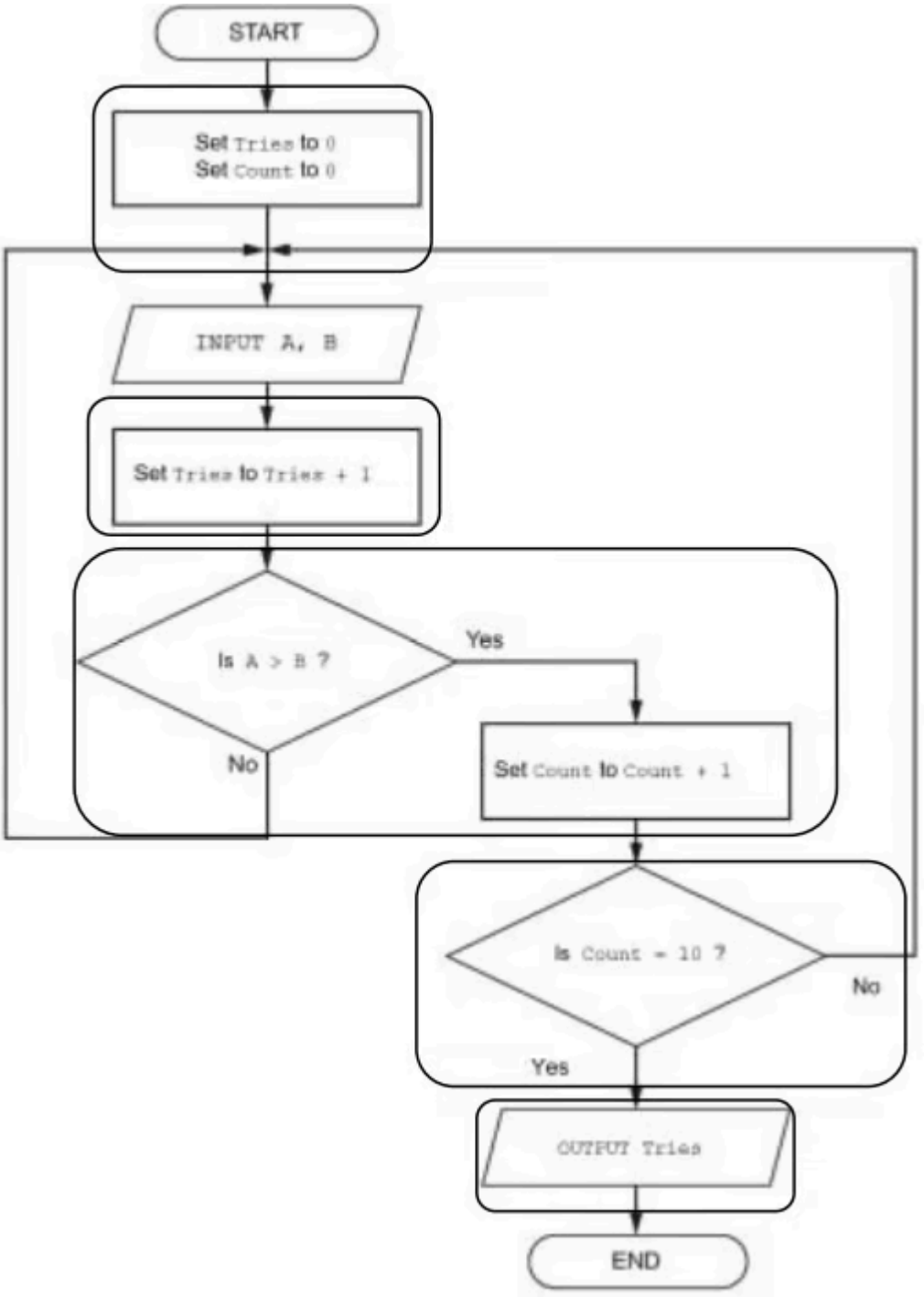
3. output the number of pairs that were input.

Complete the program flowchart.



Answer

Mark Scheme and Guidance



One mark per outlined region:

1. Initialise both counts

2. Increment Tries every time a pair is input
3. Compare A > B **and** increment Count if TRUE
4. Test for Count = 10 (10th time A > B) – MUST include Yes / No labels
5. If so output Tries, otherwise loop

(5 marks)

8 A global 1D array of strings contains three elements which are assigned values as shown:

```
Data[1] ← "aaaaaa"
Data[2] ← "bbbbbb"
Data[3] ← "cccccc"
```

Procedure Process() manipulates the values in the array.

The procedure is written in pseudocode as follows:

```
PROCEDURE Process(Format : STRING)
DECLARE Count, Index, L : INTEGER
DECLARE Result : STRING
DECLARE C : CHAR

Result ← "****"

FOR Count 1 TO LENGTH(Format) STEP 2
C ← MID(Format, Count, 1)
L ← STR_TO_NUM(MID(Format, Count + 1, 1))

Index ← (Count + 1) DIV 2

CASE OF C
'X' : Result ← TO_UPPER(Data[Index])
'Y' : Result ← TO_LOWER(Data[Index])
'Z' : Result ← "**" & Data[Index]
ENDCASE

Data[Index] ← LEFT(Result, L)
NEXT Count

ENDPROCEDURE
```

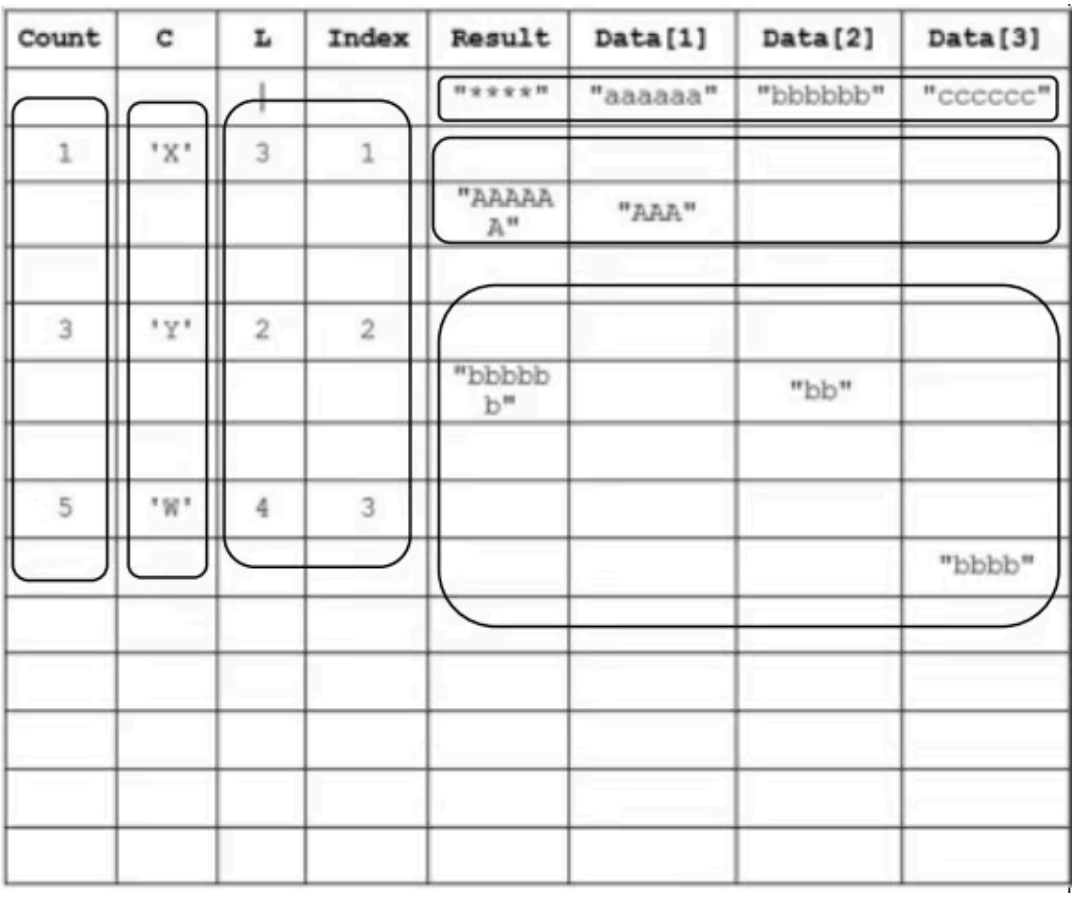
Complete the trace table by dry running the procedure when it is called as follows:

```
CALL Process("X3Y2W4")
```

Count	C	L	Index	Result	Data[1]	Data[2]	Data[3]

Answer

Mark Scheme and Guidance



Count	C	L	Index	Result	Data[1]	Data[2]	Data[3]
				"****"	"aaaaaa"	"bbbbbb"	"cccccc"
1	'X'	3	1				
				"AAAAAA"	"AAA"		
3	'Y'	2	2				
				"bbbbbb"		"bb"	
5	'W'	4	3				
							"bbbb"

One mark per zone.

(6 marks)

9 A fitness club has a computerised membership system. The fitness club offers a number of different exercise classes.

The following information is stored for each club member: name, home address, email address, mobile phone number, date of birth and the exercise(s) they are interested in.

When an exercise class is planned, a new module will send personalised text messages to each member who has expressed an interest in that exercise. Members wishing to join the class send a text message back. Members may decide not to receive future text messages by replying with the message 'STOP'.

(i) Identify **three** items of information that will be required by the new module.
Justify your choices with reference to the given scenario.

Item 1 required .....

Justification .....

Item 2 required .....

Justification .....

Item 3 required .....

Justification .....

[3]

(ii) Identify **two** operations that would be required to process data when the new module receives a text message back from a member.

Operation 1 .....

Operation 2 .....

[2]

Answer

Mark Scheme and Guidance

(i)

One mark per item **and** justification

Item: mobile phone number
Justification: to send the text message

Item: name
Justification: to personalise the text message

Item: exercise interest
Justification: to determine whether this member would be interested

(ii)

Examples include:

- Add a member to a list of those interested in the new class
- Remove the member from future SMS messages
- Read/process Message
- Identify who from

One mark for each.

Max 2 marks

(5 marks)

10 (a) Refer to the insert for the list of pseudocode functions and operators.

The following table contains pseudocode examples.

Each example may contain statements that relate to one or more of the following:

- selection
- iteration (repetition)
- input/output.

Complete the table by placing **one or more** ticks (✓) in each row.

Pseudocode example	Selection	Iteration	Input/Output
FOR Index ← 1 TO 10 Data[Index] ← 0 NEXT Index			
WRITEFILE ThisFile, "****"			
UNTIL Level > 25			

IF Mark > 74 THEN
READFILE OldFile, Data
ENDIF			

Answer

Pseudocode example	Selection	Iteration	Input/Output
FOR Index ← 1 TO 10 Data[Index] ← 0 NEXT Index		✓	
WRITEFILE ThisFile, "****"			✓
UNTIL Level > 25		✓	

IF Mark > 74 THEN
READFILE OldFile, Data
ENDIF
	✓		✓

One mark per row.

(4 marks)

(b) Program variables have data types as follows:

Variable	Data type
MyChar	CHAR
MyString	STRING
MyInt	INTEGER

The variables given in part **(b)** are chosen during the design stage of the program development life cycle.

The choices are to be documented to simplify program maintenance.

State a suitable way of documenting the variables **and** give **one** piece of information that should be recorded, in addition to the data type.

Answer

Mark Scheme and Guidance

1 mark for stating a suitable way of documenting:

- Identifier table
  1 mark for giving one piece of information that should be recorded.
  examples include:
  Explanation of what (each) variable is used for
  The purpose of (each) variable
  An example of data values stored // Initialisation value

(2 marks)

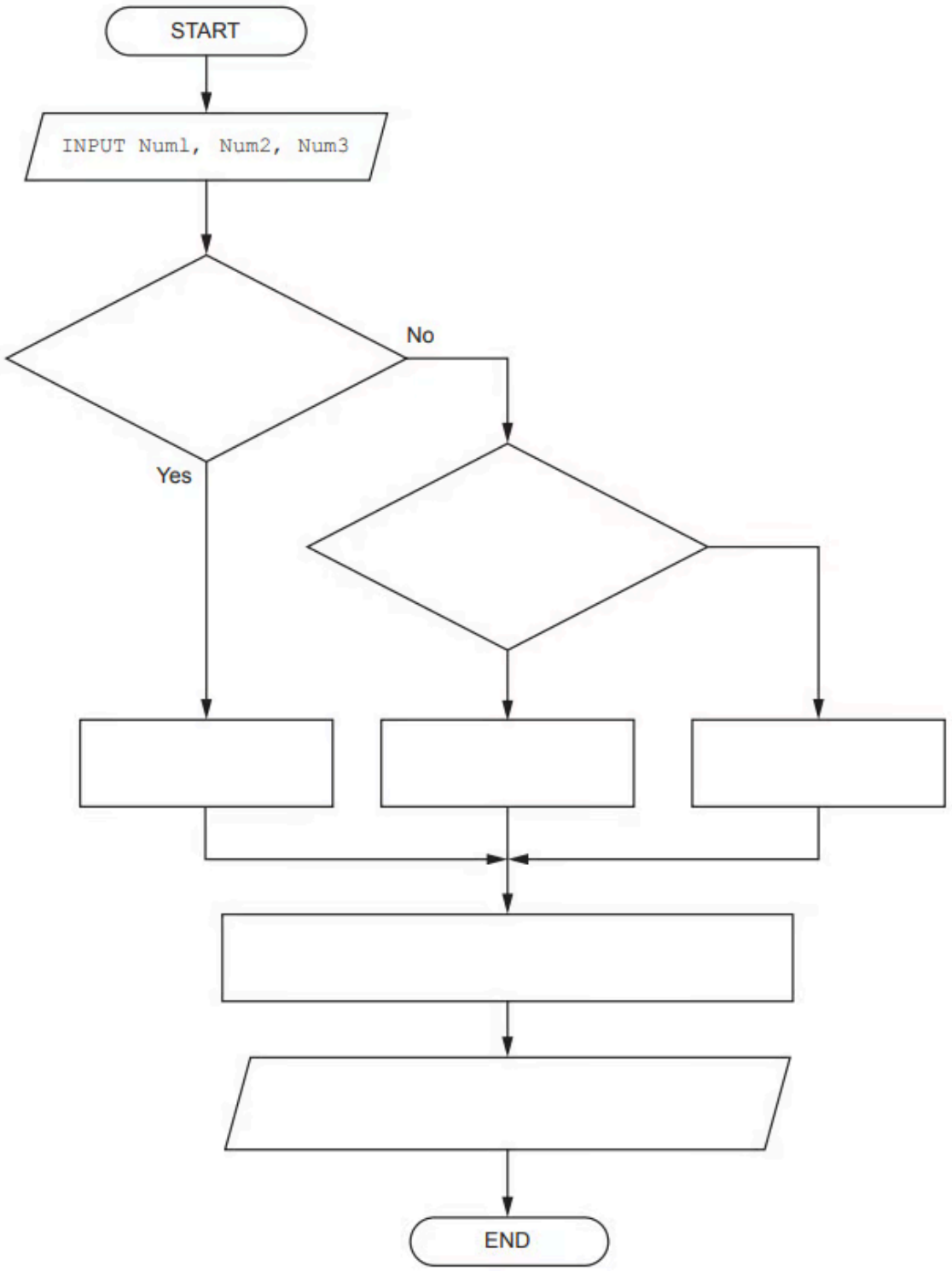
11 (a) A program is being developed.

An algorithm for part of the program will:

- input three numeric values and assign them to identifiers Num1, Num2 and Num3
- assign the largest value to variable Ans
- output a message giving the largest value and the average of the three numeric values.

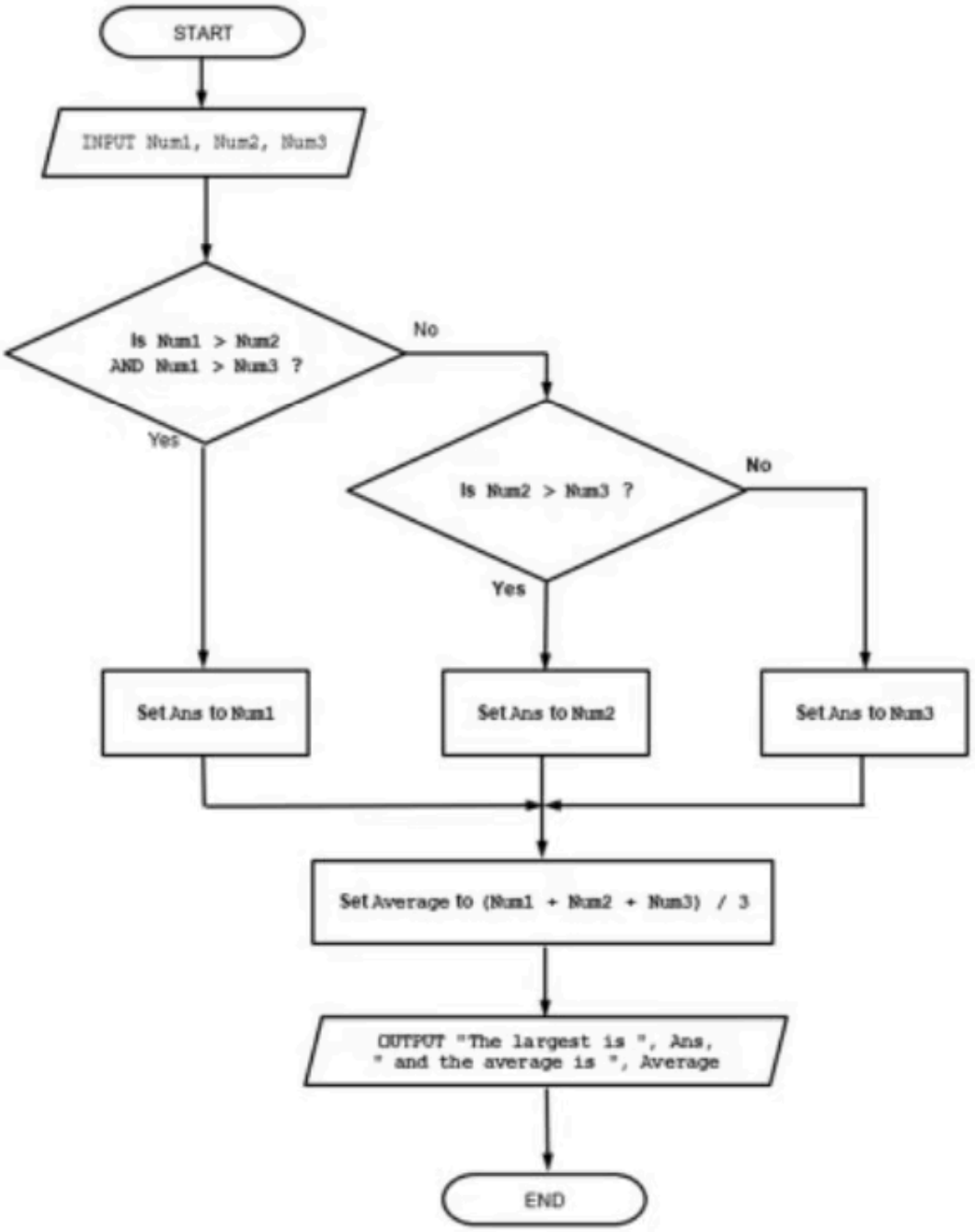
Assume the values are all different and are input in no particular order.

Complete the program flowchart on page 5 to represent the algorithm.



Answer

Mark Scheme and Guidance



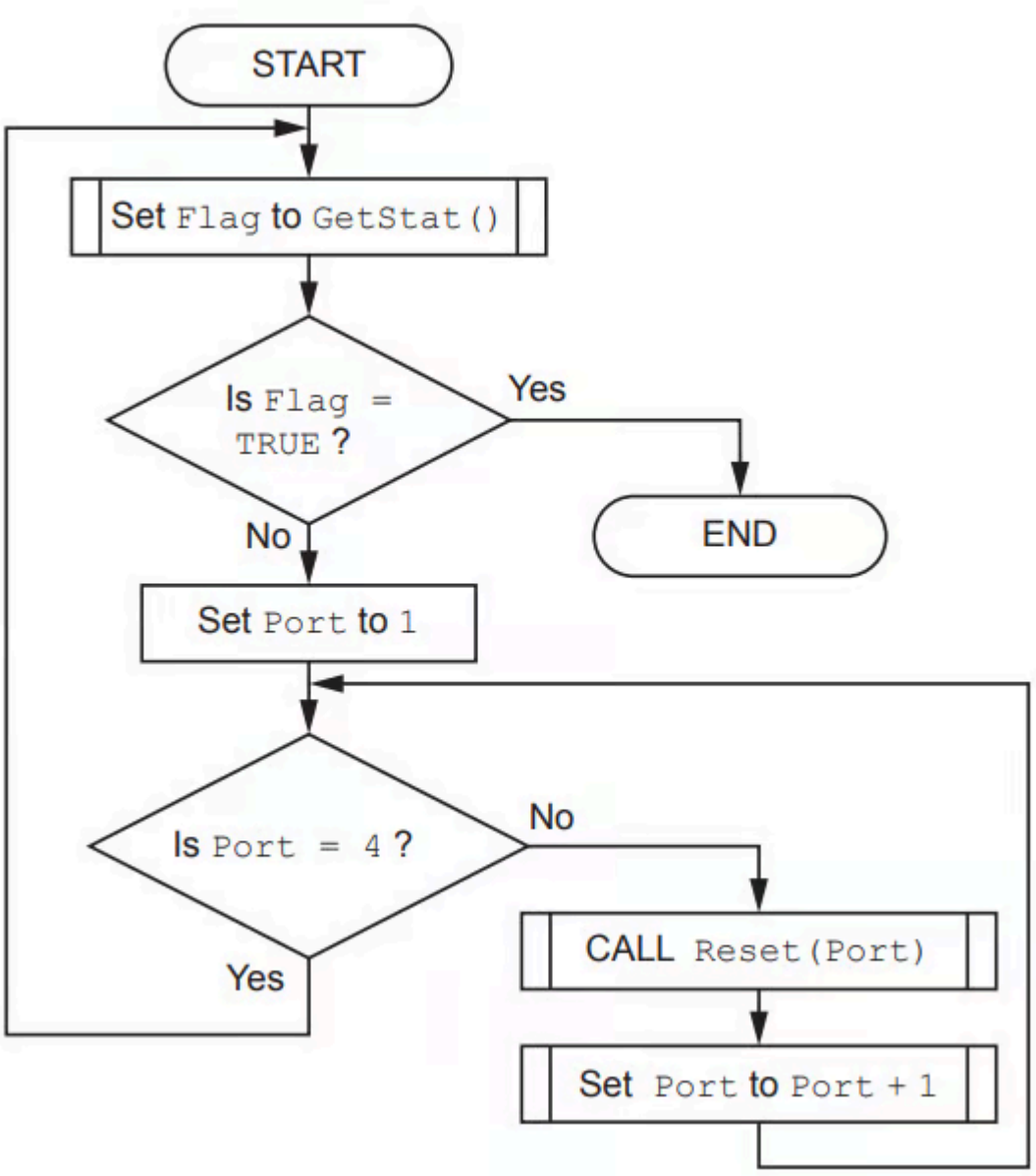
Mark points

1. Condition for selecting one of the input numbers as largest value
2. ... and assign to Ans
3. Condition for selecting largest number for all three number input and assigning to Ans

4. Average calculated using Num1, Num2 and Num3 and stored in a variable. Reject use of DIV
5. Output of Ans and average in output symbol / parallelogram

(5 marks)

(b) A different part of the program contains an algorithm represented by the following program flowchart:



Write pseudocode for the algorithm.

Answer

Mark Scheme and Guidance

Example solutions:

```
Flag ← GetStat()
WHILE Flag <> TRUE
FOR Port ← 1 TO 3
CALL Reset(Port)
NEXT Port
Flag ← GetStat()
ENDWHILE
```

Alternative:

```
REPEAT
Flag ← GetStat()
IF Flag <> TRUE THEN
FOR Port ← 1 TO 3
CALL Reset(Port)
NEXT Port
ENDIF
UNTIL Flag = TRUE
```

One mark per point:

1. (Outer) conditional loop testing Flag
2. Correct assignment of Flag from GetStat() **in a loop**
3. (Inner) loop checking / counting port // Check if Port is different to 4
4. … loop for 3 iterations
5. a call to Reset()**in a loop**

(5 marks)

12 A module InRange() will:

- be called with an integer parameter representing an index value of a record in the Batch array

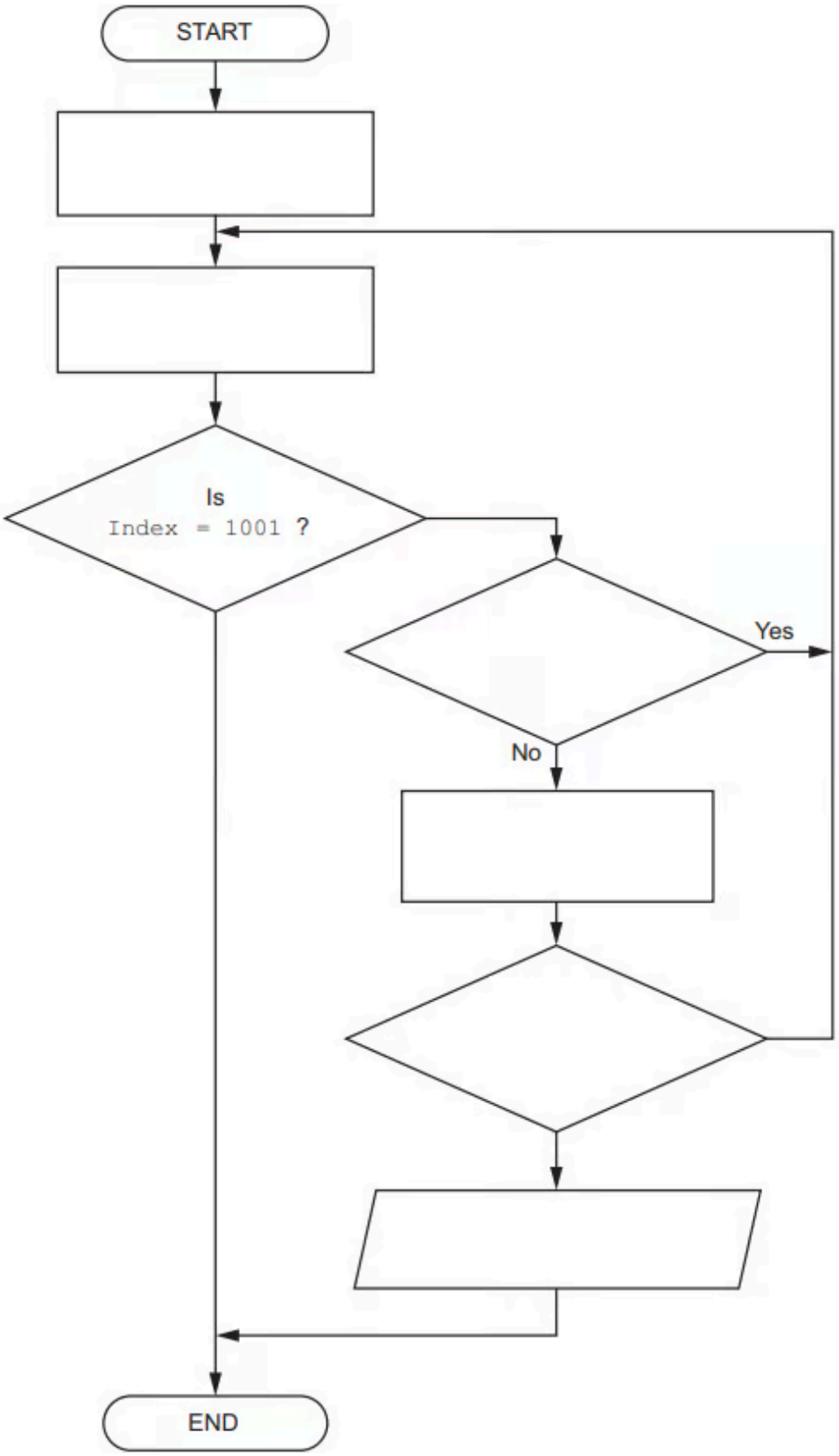
- check if the weight of the indexed component is within the acceptable range

- return TRUE if the weight is in the range and FALSE if it is **not**.

A module BatchCheck() will:

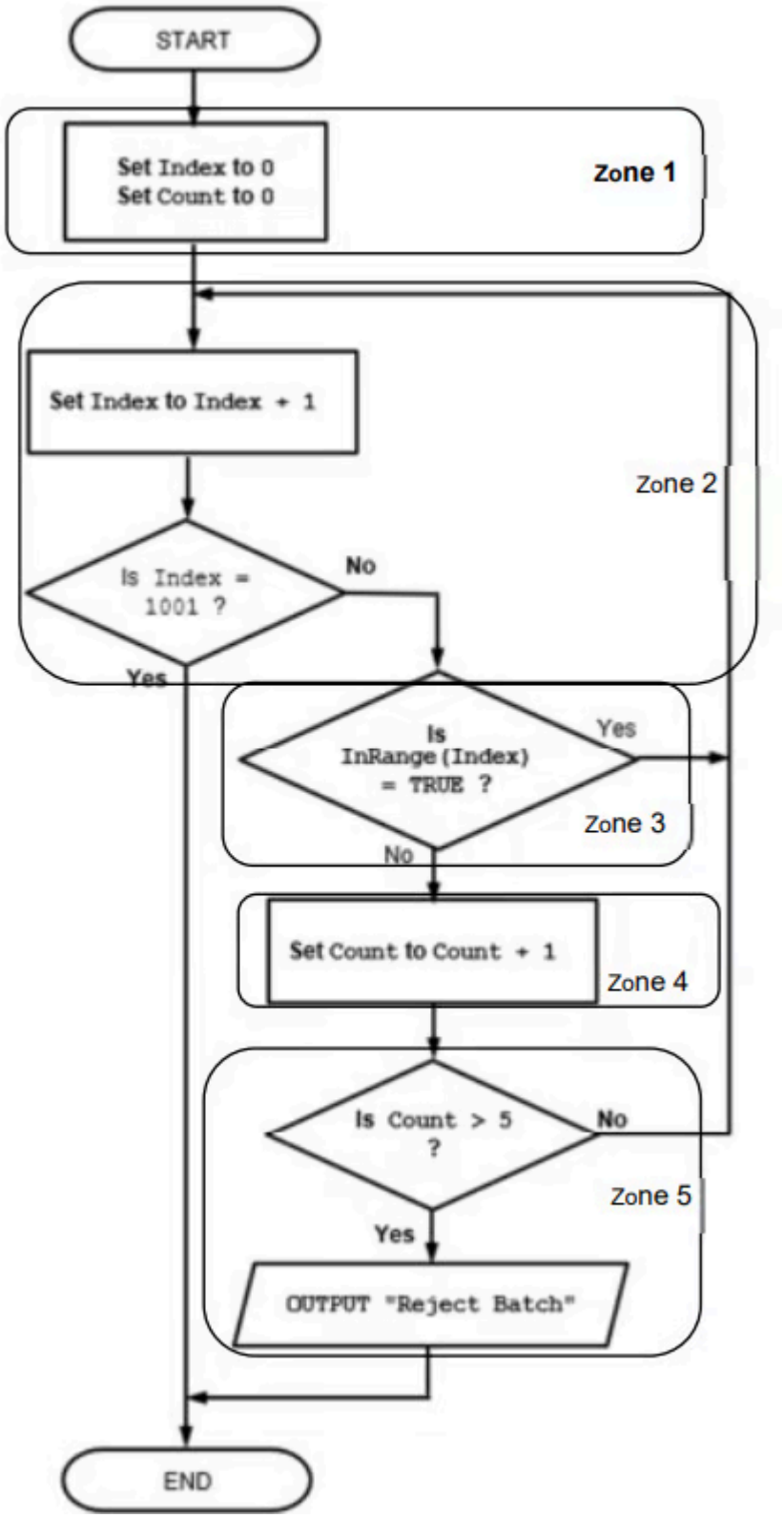
- iterate through a batch of 1000 component records
- call module InRange() to check each individual component record
- keep a count of the number of components that fail
- output a suitable warning message and immediately stop if the number of failed components exceeds 5.

Complete the program flowchart to represent the algorithm for module BatchCheck().



Answer

Mark Scheme and Guidance



One mark per zone

(5 marks)

13 A program is being designed in pseudocode.

The program contains the following declaration:

```
DECLARE Data : ARRAY[1:1000] OF STRING
```

A procedure ArrayInitialise() is written to initialise the values in the array:

```
PROCEDURE ArrayInitialise(Label : STRING)
DECLARE
Index : INTEGER Index ← 1
WHILE Index <= 1000
CASE OF (Index MOD 2)
0 : Data[Index] ← FormatA(Label)
Index ← Index + 1
1 : Data[Index] ← FormatB(Label)
Index ← Index + 1
ENDCASE
ENDWHILE
ENDPROCEDURE
```

Functions FormatA() and FormatB() apply fixed format case changes to the parameter string.

The algorithm calls one of the functions FormatA() and FormatB() each time within the loop.

Explain why this is **not** efficient **and** suggest a more efficient solution.

Answer

Mark Scheme and Guidance

Two mark for Statement of Problem:

1. The functions will return the same value every time they are called
2. … because Label / the parameter value does not change within the loop

Two marks for Solution:

3. Assign FormatA(Label) and FormatB(Label) to two (local) variables before the loop

4. Use the (new) variables in place of the function calls / in the loop // Replace the references to FormatA() and FormatB()in the CASE clauses with the new variable names

(4 marks)

14 An algorithm is expressed as follows:

- input 100 numbers, one at a time
- keep a total of all numbers input that have a value between 30 and 70 inclusive and output this total after the last number has been input.

Outline, using stepwise refinement, the five steps for this algorithm which could be used to produce pseudocode.

Do **not** use pseudocode statements in your answer.

Step 1 .....

Step 2 .....

Step 3 .....

Step 4 .....

Step 5 .....

Answer

Mark Scheme and Guidance

Max 5 marks

MP1 Set total to zero
MP2 Input a number
MP3 Check if number greater than 29 and less than 71
MP4 ... if check is true - add number to total
MP5 Repeat from step 2 99 times // for a total of 100 iterations
MP6 Output the total

(5 marks)

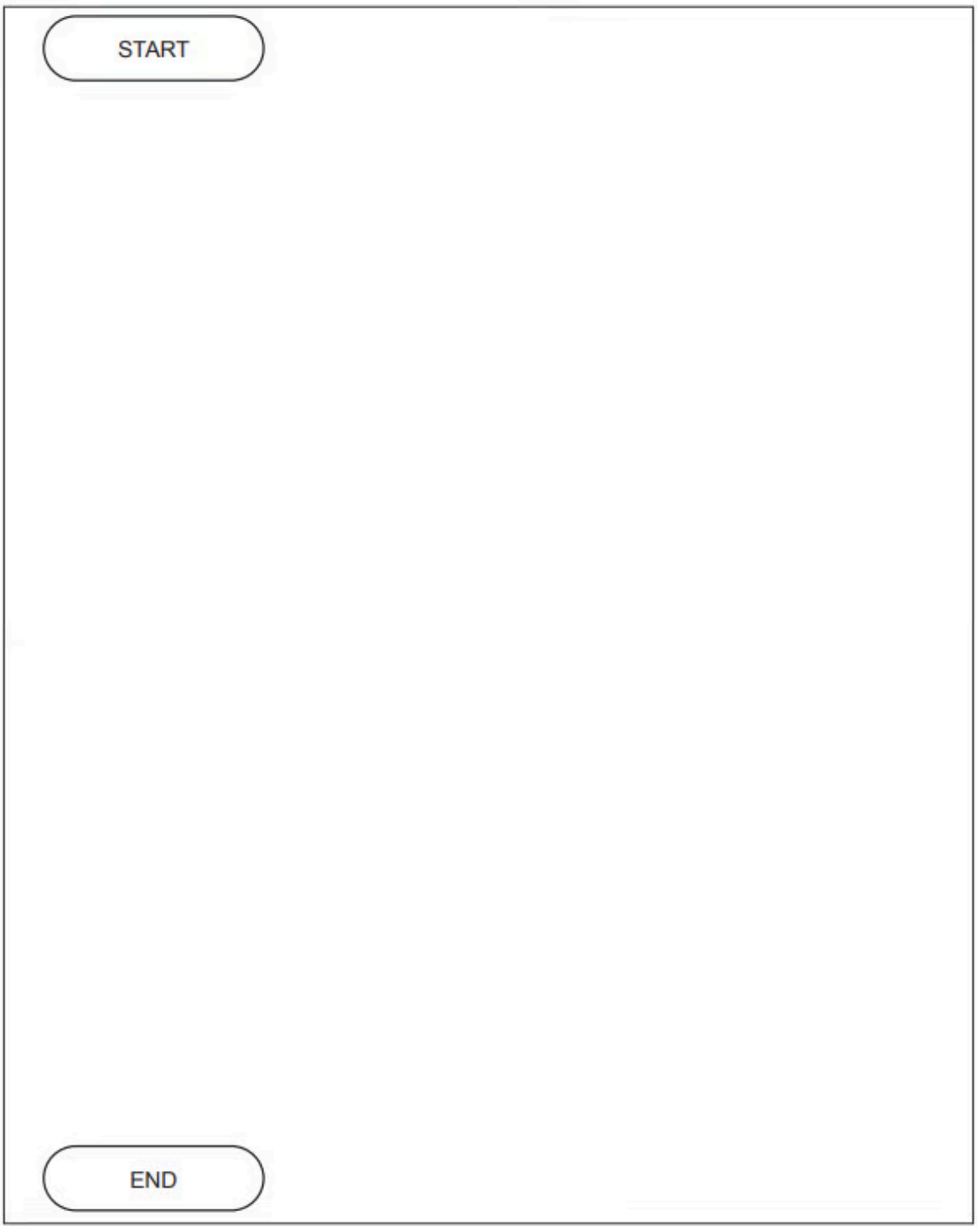
15 An algorithm will:

1. input a sequence of integer values, one at a time

2. ignore all values until the value 27 is input, then sum the remaining values in the sequence

3. stop summing values when the value 0 is input and then output the sum of the values.

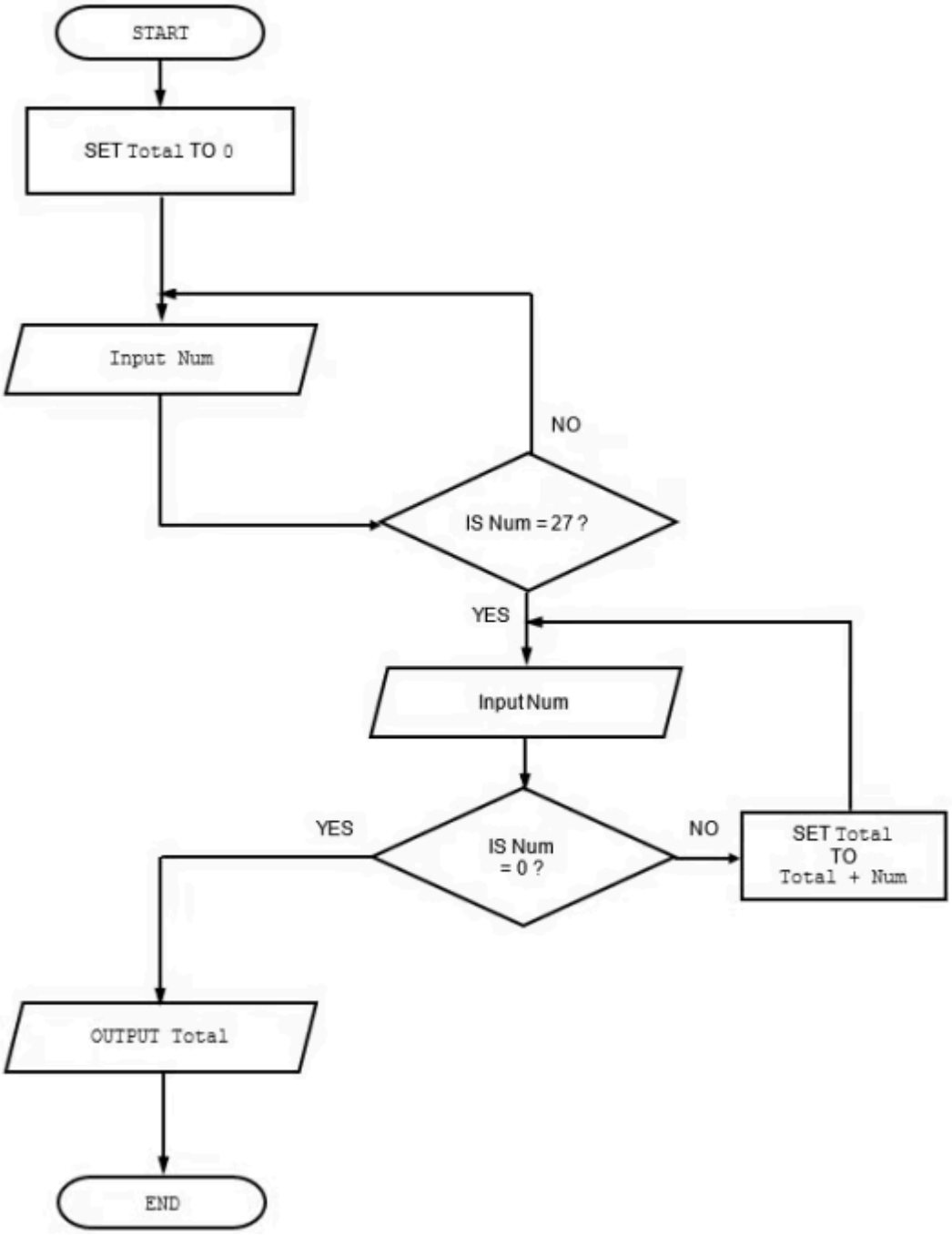
Draw a program flowchart to represent the algorithm.



Answer

Mark Scheme and Guidance

Example Solution



One mark per point:

1. Initialise Total to zero
2. Check for only **first** input of 27 **in a loop** then attempt to sum values
3. Loop until 0 input
4. Sum values **input** (after input of first 27) **in a loop**

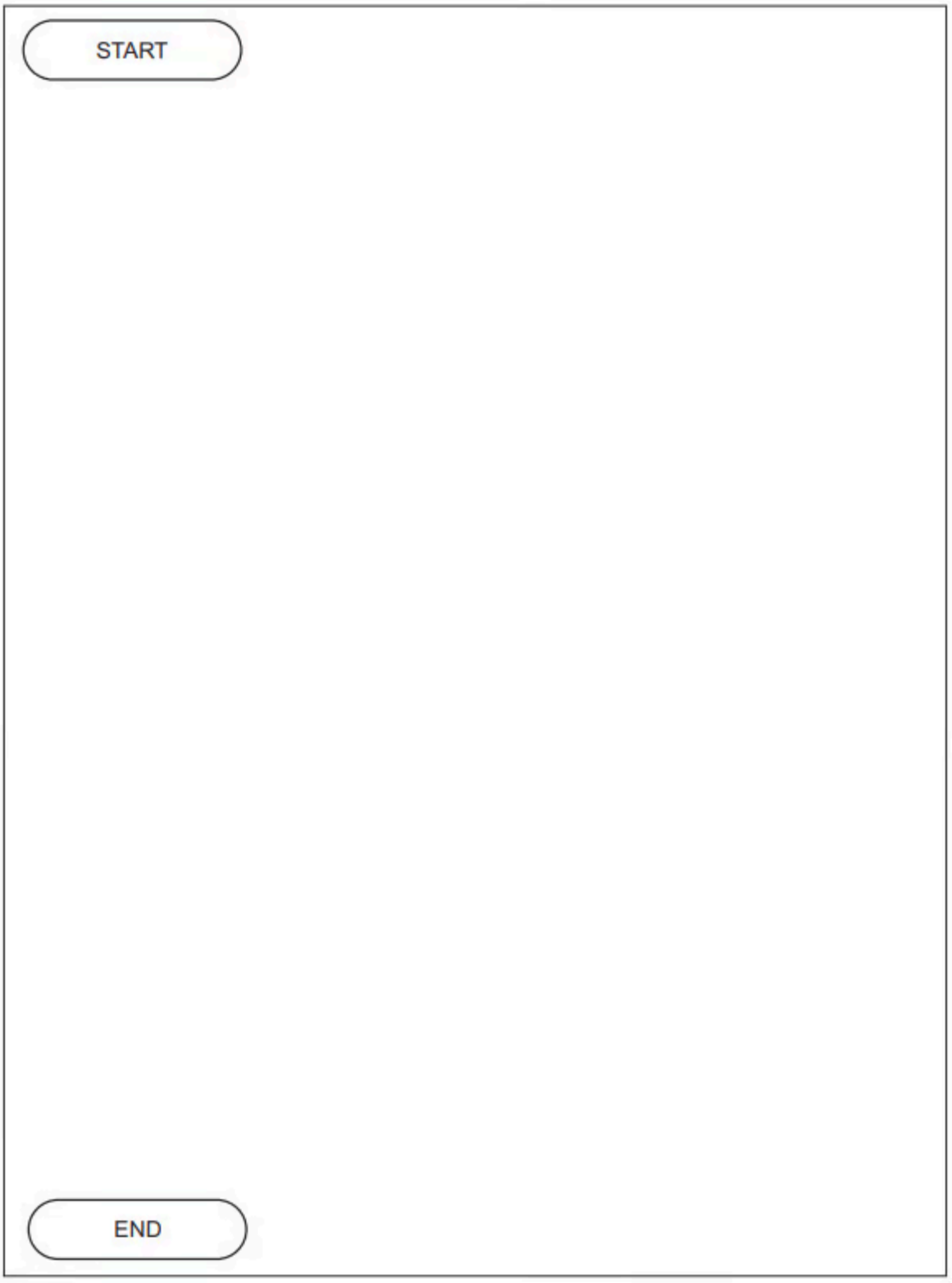
5. Output Total after a reasonable attempt

(5 marks)

16 Data is a 1D array of integers, containing 30 elements. All element values are unique.

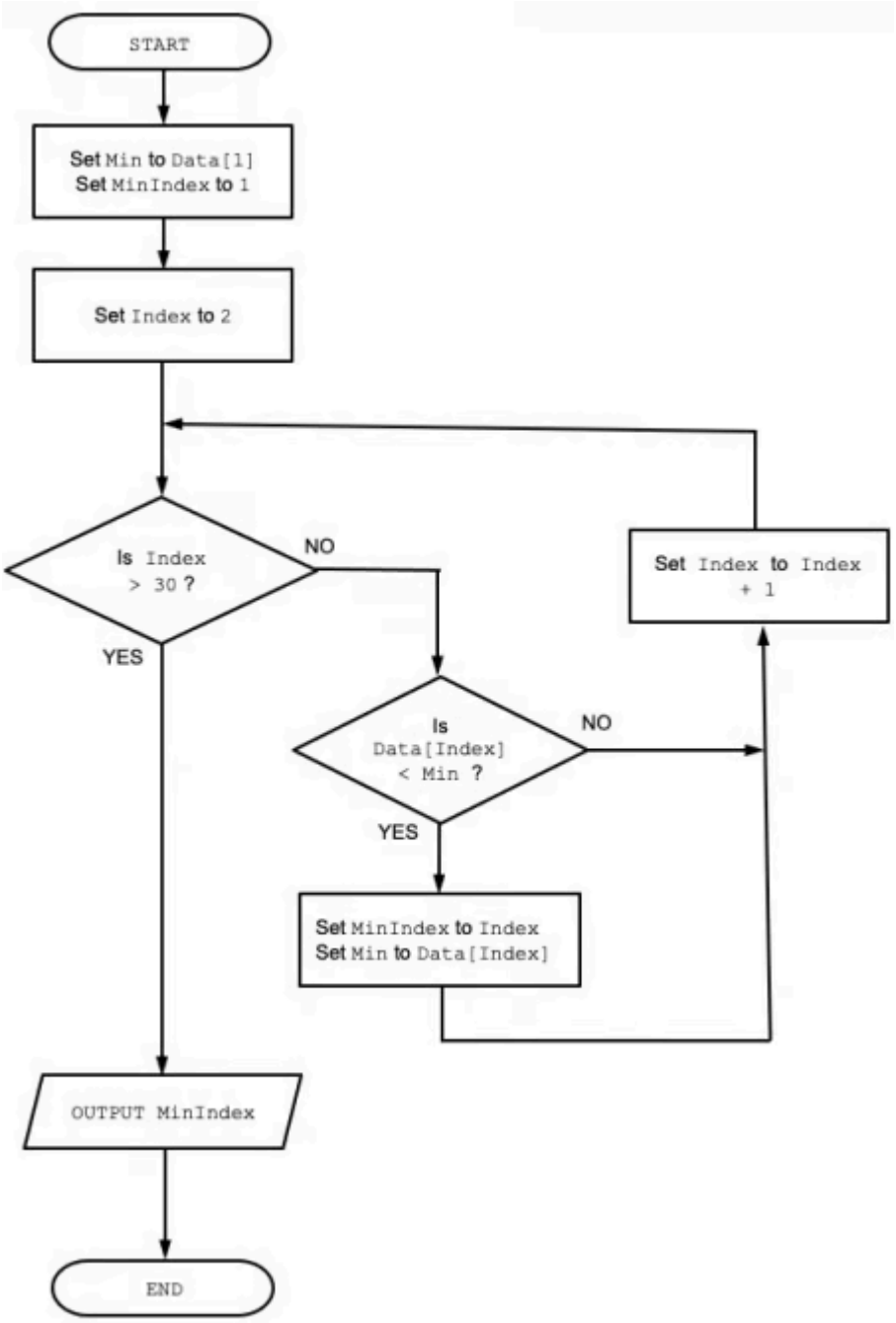
An algorithm will output the index of the element with the smallest value.

Draw a program flowchart to represent the algorithm.



Answer

Mark Scheme and Guidance

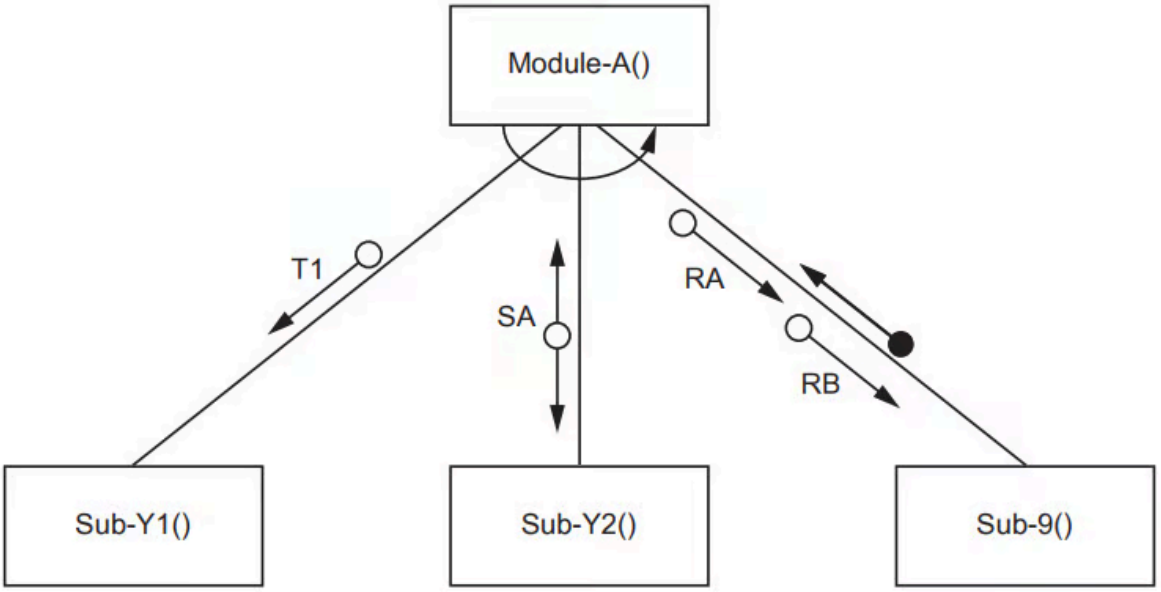


MP1 Initialise Min to first value in Data **and** MinIndex to 1
MP2 Loop through 29 more values (or 30 values in total)
MP3 Compare element from Data[] with Min

MP4 Set new Min **AND** save MinIndex when element value < Min **in a loop**
MP5 Output MinIndex

(5 marks)

17 A structure chart shows the modular structure of a program:



Explain the meaning of the curved arrow symbol which begins and ends at Module-A().

Answer

Mark Scheme and Guidance

MP1 iteration / looping
MP2 naming all four modules correctly in the correct sequence // e.g. Module-A repeatedly calls Sub-Y1,then SubY2 then Sub-9

(2 marks)