4 Study the following pseudocode. Line numbers are for reference only.

```
10 FUNCTION Convert(Name : STRING) RETURNS STRING
11
12   DECLARE Flag: BOOLEAN
13   DECLARE Index : INTEGER
14   DECLARE ThisChar : CHAR
15   DECLARE NewName : STRING
16
17   CONSTANT SPACECHAR = ' '
18
19   Flag ← TRUE
20   Index ← 1
21   NewName ← ""      // formatted name string
22
23   WHILE Index <= LENGTH(Name)
24      ThisChar ← MID(Name, Index, 1)
25      IF Flag = TRUE THEN
26         NewName ← NewName & UCASE(ThisChar)
27         IF ThisChar <> SPACECHAR THEN
28            Flag ← FALSE
29         ENDIF
30      ELSE
31         NewName ← NewName & ThisChar
32      ENDIF
33      IF ThisChar = SPACECHAR THEN
34         Flag ← TRUE
35      ENDIF
36      Index ← Index + 1
37   ENDWHILE
38
39   RETURN NewName
40
41 ENDFUNCTION
```

(a) Complete the trace table below by dry running the function when it is called as follows:

```
Result ← Convert("∇in∇a∇∇Cup")
```

Note: The symbol `'∇'` has been used to represent a space character.
Use this symbol for any space characters in the trace table.

The first row has been completed for you.

Name	Flag	Index	NewName	ThisChar
"∇in∇a∇∇Cup"				

[5]

(b) The pseudocode for `Convert()` contains a conditional loop.

State a more appropriate loop structure.

Justify your answer.

Loop structure .....

.....

Justification .....

.....

.....

[2]

(c) Two changes need to be made to the algorithm.

Change 1: Convert to lower case any character that is not the first character after a space.
Change 2: Replace multiple spaces with a single space.

(i) Change 1 may be implemented by modifying one line of the pseudocode.

Write the modified line.

.....

.....[1]

(ii) Change 2 may be implemented by moving one line of the pseudocode.

Write the number of the line to be moved and state its new position.

Line number .....

New position .....

.....

[2]

5 **(a)** A student is learning about arrays.

She wants to write a program to:

- declare a 1D array `RNum` of 100 elements of type `INTEGER`
- assign each element a random value in the range 1 to 200 inclusive
- count and output how many numbers generated were between 66 and 173 inclusive.

(i) Write pseudocode to represent the algorithm.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [6]

(ii) The student decides to modify the algorithm so that each element of the array will contain a unique value.

Describe the changes that the student needs to make to the algorithm.

.....

.....

.....

.....

.....

..... [3]

[Turn over

(b) The following is a pseudocode function.

Line numbers are given for reference only.

```
01  FUNCTION StringClean(InString : STRING) RETURNS STRING
02
03     DECLARE NextChar : CHAR
04     DECLARE OutString : STRING
05     DECLARE Counter : INTEGER
06
07     OutString ← ""
08
09     FOR Counter ← 1 TO LENGTH(InString)
10        NextChar ← MID(InString, Counter, 1)
11        NextChar ← LCASE(NextChar)
12        IF NOT((NextChar < 'a') OR (NextChar > 'z')) THEN
13           OutString ← OutString & NextChar
14        ENDIF
15     NEXT Counter
16
17     RETURN OutString
18
19  ENDFUNCTION
```

(i) Examine the pseudocode and complete the following table.

	Answer
Give a line number containing an example of an initialisation statement.	
Give a line number containing the start of a repeating block of code.	
Give a line number containing a logic operation.	
Give the number of parameters to the function `MID()`.	

[4]

(ii) Write a simplified version of the statement in line `12`.

.....

..... [2]

6 **(a)** An algorithm will:

- output **each** integer value between 100 and 200 that ends with the digit 7, for example, 107
- output a final count of the number of values that are output.

Write pseudocode for this algorithm.

Any variables used must be declared.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [5]

(b) Study the following pseudocode.

```
CASE OF MySwitch
   1: ThisChar ← 'a'
   2: ThisChar ← 'y'
   3: ThisChar ← '7'
   OTHERWISE: ThisChar ← '*'
ENDCASE
```

Write pseudocode with the same functionality **without** using a `CASE` structure.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [4]

5 Study the following pseudocode. Line numbers are for reference only.

```
10 PROCEDURE Encode()
11   DECLARE CountA, CountB, ThisNum : INTEGER
12   DECLARE ThisChar : CHAR
13   DECLARE Flag : BOOLEAN
14   CountA ← 0
15   CountB ← 10
16   Flag ← TRUE
17   INPUT ThisNum
18   WHILE ThisNum <> 0
19      ThisChar ← LEFT(NUM_TO_STR(ThisNum), 1)
20      IF Flag = TRUE THEN
21         CASE OF ThisChar
22            '1' : CountA ← CountA + 1
23            '2' : IF CountB < 10 THEN
24                     CountA ← CountA + 1
25                  ENDIF
26            '3' : CountB ← CountB - 1
27            '4' : CountB ← CountB - 1
28                  Flag ← FALSE
29            OTHERWISE : OUTPUT "Ignored"
30         ENDCASE
31      ELSE
32         IF CountA > 2 THEN
33            Flag ← NOT Flag
34            OUTPUT "Flip"
35         ELSE
36            CountA ← 4
37         ENDIF
38      ENDIF
39      INPUT ThisNum
40   ENDWHILE
41   OUTPUT CountA
42 ENDPROCEDURE
```

(a) Procedure `Encode()` contains a loop structure.

Identify the type of loop **and** state the condition that ends the loop.

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

Type .....

Condition .....

.....

[2]

(b) Complete the trace table below by dry running the procedure `Encode()` when the following values are input:

`12, 24, 57, 43, 56, 22, 31, 32, 47, 99, 0`

The first row is already complete.

ThisNum	ThisChar	CountA	CountB	Flag	OUTPUT
		0	10	TRUE	

[6]

(c) Procedure `Encode()` is part of a modular program. Integration testing is to be carried out on the program.

Describe **integration testing**.

.....

.....

.....

..... [2]

7 The following pseudocode represents an algorithm intended to output the last three lines as they appear in a text file. Line numbers are provided for reference only.

```
10  PROCEDURE LastLines(ThisFile : STRING)
11     DECLARE ThisLine : STRING
12     DECLARE Buffer : ARRAY[1:3] OF STRING
13     DECLARE LineNum : INTEGER
14     LineNum ← 1
15     OPENFILE ThisFile FOR READ
16
17     WHILE NOT EOF(ThisFile)
18        READFILE Thisfile, ThisLine // read a line
19        Buffer[LineNum] ← ThisLine
20        LineNum ← LineNum + 1
21        IF LineNum = 4 THEN
22           LineNum ← 1
23        ENDIF
24     ENDWHILE
25
26     CLOSEFILE ThisFile
27     FOR LineNum ← 1 TO 3
28        OUTPUT Buffer[LineNum]
29     NEXT LineNum
30  ENDPROCEDURE
```

(a) There is an error in the algorithm. In certain cases, a text file will have at least three lines but the output will be incorrect.

(i) State how the output may be incorrect.

.....

..... [1]

(ii) Describe the error in the algorithm **and** explain how it may be corrected.

Description .....

.....

.....

.....

Explanation .....

.....

.....

.....

[4]

(b) The original algorithm is implemented and sometimes the last three lines of the text file are output correctly.

State the condition that results in the correct output.

.....

..... [1]

(c) Lines 20 to 23 inclusive could be replaced with a single pseudocode statement.

Write the pseudocode statement.

.....

..... [2]

4 The following is a procedure design in pseudocode.

Line numbers are given for reference only.

```
10  PROCEDURE Check(InString : STRING)
11     DECLARE Odds, Evens, Index : INTEGER
12
13     Odds ← 0
14     Evens ← 0
15     Index ← 1
16
17     WHILE Index <= LENGTH(InString)
18        IF STR_TO_NUM(MID(InString, Index, 1)) MOD 2 <> 0 THEN
19           Odds ← Odds + 1
20        ELSE
21           Evens ← Evens + 1
22        ENDIF
23        Index ← Index + 1
24     ENDWHILE
25
26     CALL Result(Odds, Evens)
27  ENDPROCEDURE
```

(a) Complete the following table by giving the answers, using the given pseudocode.

	Answer
A line number containing a variable being incremented	
The type of loop structure	
The number of functions used	
The number of parameters passed to `STR_TO_NUM()`	
The name of a procedure other than `Check()`	

[5]

(b) The pseudocode includes several features that make it easier to read and understand.

Identify **three** of these features.

1 .....

2 .....

3 .....

[3]

(c) (i) The loop structure used in the pseudocode is not the most appropriate.

State a more appropriate loop structure **and** justify your choice.

Loop structure .....

Justification .....

.....

.....
[2]

(ii) The appropriate loop structure is now used. Two lines of pseudocode are changed and two lines are removed.

Write the line numbers of the two lines that are removed.

.....

..... [1]

6 The following pseudocode algorithm attempts to check whether a string is a valid email address.

```
FUNCTION IsValid(InString : STRING) RETURNS BOOLEAN
   DECLARE Index, Dots, Ats, Others : INTEGER
   DECLARE NextChar : CHAR
   DECLARE Valid : BOOLEAN

   Index ← 1
   Dots ← 0
   Ats ← 0
   Others ← 0
   Valid ← TRUE

   REPEAT
      NextChar ← MID(InString, Index, 1)
      CASE OF NextChar
         '.' : Dots ← Dots + 1
         '@' : Ats ← Ats + 1
               IF Ats > 1 THEN
                  Valid ← FALSE
               ENDIF
         OTHERWISE : Others ← Others + 1
      ENDCASE

      IF Dots > 1 AND Ats = 0 THEN
         Valid ← FALSE
      ELSE
         Index ← Index + 1
      ENDIF

   UNTIL Index > LENGTH(InString) OR Valid = FALSE

   IF NOT (Dots >= 1 AND Ats = 1 AND Others > 8) THEN
      Valid ← FALSE
   ENDIF

   RETURN Valid

ENDFUNCTION
```

(a) Part of the validation is implemented by the line:

```
IF NOT (Dots >= 1 AND Ats = 1 AND Others > 8) THEN
```

State the values that would result in the condition evaluating to `TRUE`.

.....

.....

..... [1]

(b) (i) Complete the trace table by dry running the function when it is called as follows:

```
Result ← IsValid("Liz.123@big@net")
```

Index	NextChar	Dots	Ats	Others	Valid

[5]

(ii) State the value returned when `IsValid()` is called using the expression shown in **part (b)(i)**.

..... [1]

5 Examine the following pseudocode.

```
IF A = TRUE THEN
   IF B = TRUE THEN
      IF C = TRUE THEN
         CALL Sub1()
      ELSE
         CALL Sub2()
      ENDIF
   ENDIF
ELSE
   IF B = TRUE THEN
      IF C = TRUE THEN
        CALL Sub4()
      ELSE
        CALL Sub3()
      ENDIF
   ELSE
      IF C = FALSE THEN
        CALL Sub3()
      ELSE
        CALL Sub4()
      ENDIF
   ENDIF
ENDIF
```

A programmer wants to re-write the pseudocode as **four** separate `IF...THEN...ENDIF` statements, each containing a single `CALL` statement. This involves writing a single, simplified logic expression as the condition in each statement.

Write the amended pseudocode.

1 .....

.....

.....

2 .....

.....

.....

3 .....

.....

.....

4 .....

.....

.....

[4]

5 An algorithm is designed to find the smallest numeric value from an input sequence and count how many numeric values have been input.
An example of an input sequence is:

23, AB56, 17, 23ZW, 4, 10, END

Numeric input values are all integers and non-numeric input is ignored, except for the string "END" which is used to terminate the sequence.

The algorithm is expressed in pseudocode as shown:

```
DECLARE NextInput : STRING
DECLARE Min, Count, Num : INTEGER

Min ← 999
Count ← 0

REPEAT
   INPUT NextInput
   IF IS_NUM(NextInput) = TRUE THEN
      Num ← STR_TO_NUM(NextInput)
      IF Num > Min THEN
         Min ← Num
      ENDIF
      Count ← Count & 1
   ENDIF
UNTIL NextInput ← "END"

OUTPUT "The minimum value is ", Min, " and the count was ", Count
```

(a) The pseudocode contains three errors due to the incorrect use of **operators**.

Identify each error **and** state the correction required.

1 .....

.....

2 .....

.....

3 .....

.....

[3]

(b) The operator errors are corrected and the algorithm is tested as follows:

The input sequence:

```
18, 4, ONE, 27, 189, ERIC, 3, 65, END
```

produces the output:

```
The minimum value is 3 and the count was 6
```

The algorithm is tested with a different test data sequence. The sequence contains a mix of integer and non-numeric values. It is terminated correctly but the algorithm produces unexpected results.

(i) Explain the problem with the algorithm.

.....

.....

.....

..... [2]

(ii) Give a sequence of **four** test data values that could be input to demonstrate the problem.

Value 1 .....

Value 2 .....

Value 3 .....

Value 4 .....

[2]

5 A global 1D array of integers contains four elements, which are assigned values as shown:

```
Mix[1] ← 1
Mix[2] ← 3
Mix[3] ← 4
Mix[4] ← 2
```

A procedure `Process()` manipulates the values in the array.

The procedure is written in pseudocode:

```
PROCEDURE Process(Start : INTEGER)
   DECLARE Value, Index, Count : INTEGER

   Index ← Start
   Count ← 0

   REPEAT
      Value ← Mix[Index]
      Mix[Index] ← Mix[Index] - 1
      Index ← Value
      Count ← Count + 1
   UNTIL Count = 5

   Mix[4] ← Count * Index

ENDPROCEDURE
```

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

```
CALL Process(3)
```

Index	Value	Count	Mix[1]	Mix[2]	Mix[3]	Mix[4]

[6]

5 A global 1D array of integers contains four elements, which are assigned values as shown:

```
Mix[1] ← 4
Mix[2] ← 2
Mix[3] ← 3
Mix[4] ← 5
```

A procedure `Process()` manipulates the values in the array.

The procedure is written in pseudocode as follows:

```
PROCEDURE Process(Start : INTEGER)
   DECLARE Value, Index, Total : INTEGER

   Index ← Start
   Total ← 0

   WHILE Total < 20
      Value ← Mix[Index]
      Total ← Total + Value

      IF Index < 4 THEN
         Mix[Index] ← Mix[Index] + Mix[Index+1]
      ELSE
         Mix[Index] ← Mix[Index] + Mix[1]
      ENDIF
      Index ← (Value MOD 4) + 1

   ENDWHILE

   Mix[1] ← Total * Index

ENDPROCEDURE
```

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

```
CALL Process(2)
```

Index	Value	Total	Mix[1]	Mix[2]	Mix[3]	Mix[4]

[6]