MARKING SCHEME FOR PSEUDOCODE AND ALGORITHM

1.

1(a) — **4**

Assignment statement	Data type
`A ← LEFT(MyName, 1)`	CHAR / STRING
`B ← Total * 2`	INTEGER / REAL
`C ← INT(ItemCost) / 3`	REAL
`D ← "Odd OR Even"`	STRING

One mark per row

1(b) — **4**

Expression	Evaluates to
`Tries < 10 AND NOT Sorted`	TRUE
`Tries MOD 4`	1
`TO_LOWER(MID(ID, 3, 1))`	'a' // "a"
`LENGTH(ID & "xx") >= Tries`	TRUE

One mark per row

1(c)(i) — **1**

The names do not reflect / indicate the purpose of the variable // the names are not meaningful

1(c)(ii) — **1**

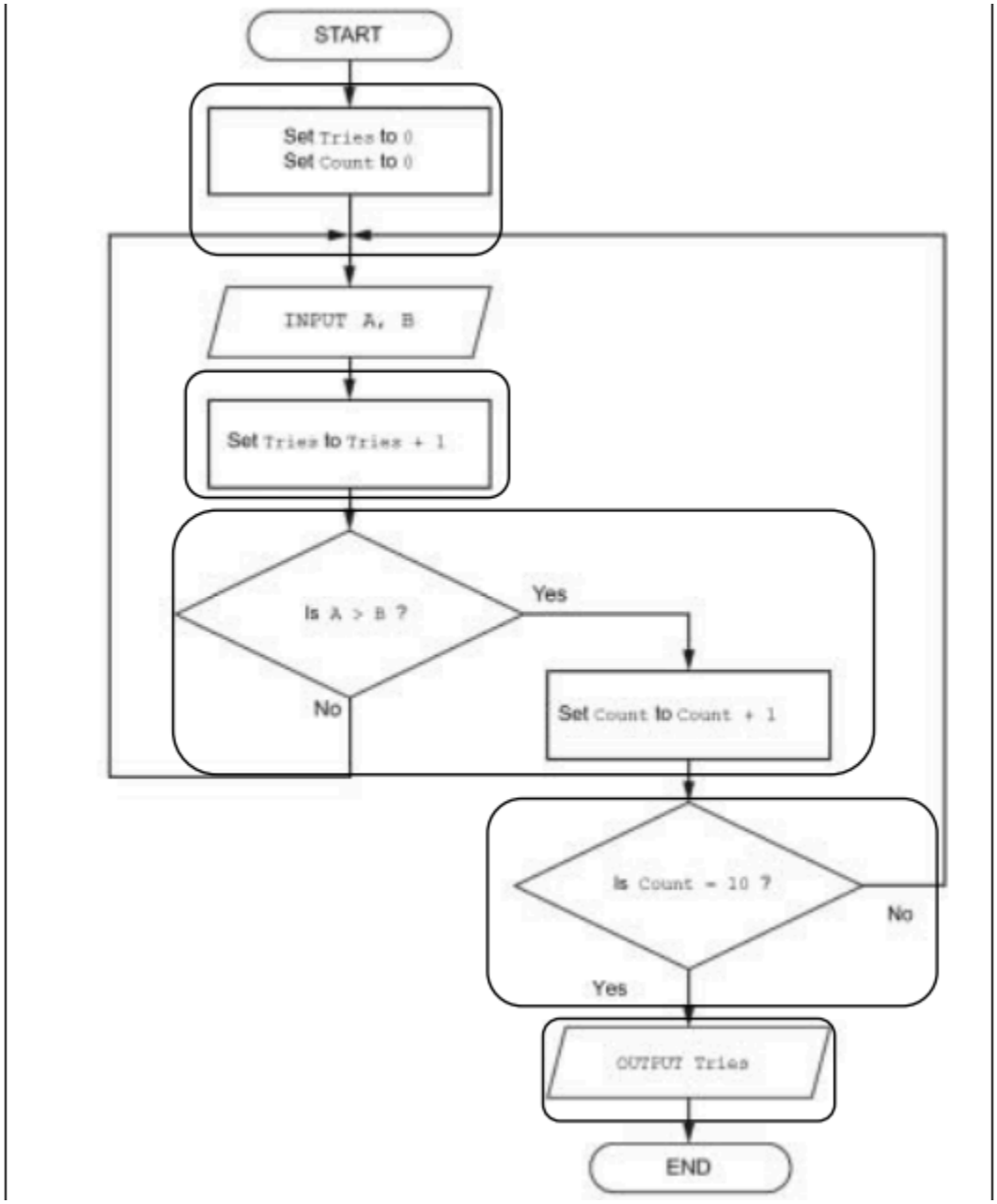
They make the program more difficult to understand / debug / maintain

1(c)(iii) — **1**

One mark from:
- Indentation / use of white space
- Capitalisation of keywords
- Use of comments
- Use of modular programming
- Use of local variables

Note: max 1 mark

2.


START
Set Tries to 0
Set Count to 0
INPUT A, B
Set Tries to Tries + 1
Is A > B ?
Yes
No
Set Count to Count + 1
Is Count = 10 ?
No
Yes
OUTPUT Tries
END

	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	
2(b)	One mark per point: 1 A (variable of type) string will be input // by example e.g. "67,72" 2 A special / identified character would need to be used to separate each numeric value // all numbers are fixed length	**2**

3.

(a) **4**

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.

(b)

Expression
`MyInt ← INT (3.1415926)`
`MyChar ← MID ("Elwood", 3, 1)`
Any of: • `MyString ← NUM_TO_STR (INT (27.509))` • `MyString ← CHR (INT (27.509))` • `MyString ← TO_UPPER( NUM_TO_STR( 27.509))` • `MyString ← TO_LOWER( NUM_TO_STR( 27.509))`
Any of: • `MyInt ← STR_TO_NUM ( RIGHT ("ABC123", 3))` • `MyInt ← LENGTH ( RIGHT ("ABC123", 3))` • `MyInt ← LENGTH ( LEFT ("ABC123", 3))`

One mark per row

4

(c)

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

4.

5

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

(b)	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

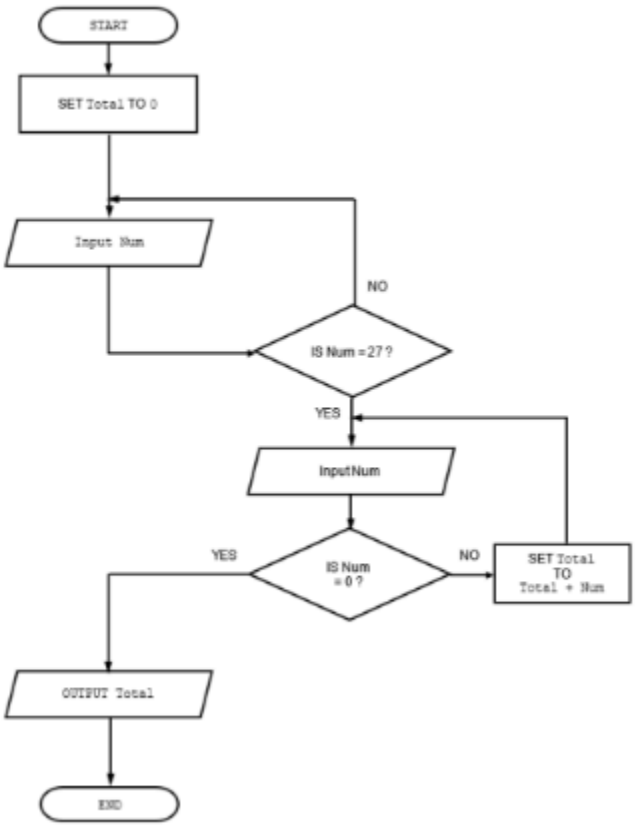
5.

2(a)	**Max 5** marks **MP1** Set total to <u>zero</u> **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**
2(b)	**MP1** An iterative construct // a (count-controlled) loop **MP2** A selection construct // an IF statement	**2**

6.

5(a)	**MP1** `Num > Min` should be `Num < Min` **MP2** `Count & 1` should be `Count + 1` **MP3** `NextInput ← "END"` should be `NextInput = "END"`	**3**
(b)(i)	**MP1** If all the numeric input values are greater than 999 // If there are no numeric values in the sequence **MP2** then the minimum will be given as **999** (and not one of the input values)	**2**
(b)(ii)	Many possible correct answers, for example: **MP1** Mixture non-numeric and numeric with 3 or 4 values - with all numerics greater than 999 Examples: 1325, DOG, 7868, 7615 // SNAKE, 3478, SPIDER **MP2** Final value: END	**2**

7.

(a)	Example Solution 	**5**

	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	
2(b)	**One** mark per point: 1 Name: (pre / post) conditional loop 2 Justification: the number of iterations is not known // loop ends following a specific input (in the loop)	**2**

8.

(a)(i)	Use of constants	**1**
(a)(ii)	One mark per bullet point (or equivalent to max 3): 1 Postal rates are entered once only 2 Avoids input error / changing the cost accidentally // avoids different values for postal rates at different points in the program 3 When required, the constant representing the postal rate value is changed once only // easier to maintain the program when the postal rates change 4 Makes the program easier to understand **Note: Max 3 marks**	**3**
1(b)	One mark per bullet point: • Indentation • White space • Comments • Sensible / meaningful variable names // use of Camel Case • Capitalised keywords **Note: Max 3 marks**	**3**
1(c)	One mark per bullet point: • BOOLEAN • REAL • STRING	**3**