

3(a)(i)	Pseudocode: <pre> TYPE RentalRecord DECLARE RentalID : STRING DECLARE CarID : INTEGER DECLARE DisCode : CHAR DECLARE Start : DATE DECLARE Duration : INTEGER DECLARE Completed : BOOLEAN ENDTYPE </pre> Mark as follows: - One mark for TYPE RentalRecord and ENDTYPE statements - One mark for RentalID and CarID declared correctly - One mark for DisCode and Start declared correctly - One mark for Duration and Completed declared correctly
3(a)(ii)	<u>DECLARE Rental : ARRAY[1:500] OF RentalRecord</u> One mark per underlined phrase
3(b)	One mark per point: Different data types held (for each rental) under a single identifier - Allows for iteration through <u>rentals/records</u> // Can access rentals/records in a loop // Can (directly) access a specific <u>rental/record</u> using <u>an</u> (array) <u>index</u> - Simplifies the code/program / less code needed / reduces code duplication // Makes code/program easier to understand // Makes testing / debugging easier // Program maintenance easier

1(a)

Pseudocode example	Selection	Iteration	Subroutine
IF Status = FALSE THEN FOR Count ← 1 TO 20 CALL Reset (Count) NEXT Count ENDIF	✓	✓	✓
OTHERWISE : Status ← TRUE	✓		
WHILE AllDone() = TRUE		✓	✓
30 : NextChar ← 'X'	✓		

One mark per row

1(b)

Variable	Example data value	Data type
Result	5.42	REAL
MonthLetter	"JFMAMJJASOND"	STRING
Birthday	15/11/2009	DATE

One mark per row

1(c)

Expression	Evaluates to
INT(Result) + 1 > 6	FALSE
NUM_TO_STR(LENGTH (MonthLetter))	"12"
NUM_TO_STR(Result + "3.2")	ERROR
MID(MonthLetter, MONTH (Birthday) - 2, 1)	"S"

One mark per row

1(a)(i)	DECLARE Member1 : ClubMember
1(a)(ii)	One mark for each correct answer Example answer Member1.Code ← 984632 Member1.FeesPaid ← TRUE
1(b)(i)	One mark per mark point (Max 2) MP1 TYPE Activity= MP2 (Badminton, Football, Golf, Snooker, Swimming, Tennis) Example answer TYPE Activity = (Badminton, Football, Golf, Snooker, Swimming, Tennis)
1(b)(ii)	DECLARE Choice : Activity

1(a)(i)	One mark for Choice of program development cycle depends on the approach/method required to produce the program Or One mark for a factor that would influence the choice of program development cycle, e.g. • Rapid development of program/prototypes required • Need for prototypes (at an early stage) • Complexity of problem • Skills / experience of development team / programmer(s) • Budget / Time / Resources available • Size of development team • Developer / programmer can return to earlier stages / any stage • Avoidance of repeating previous stages in development of program • How much involvement clients will have • Allows the (program) requirements to be changed during program development • (Program) requirements agreed at start of development of program Max 1										
1(a)(ii)	Which programming language would best be suited to control the production line // Which programming language would best suit the problem being solved										
1(b)(i)	Examples include: • Change to (production line) requirements • New technology / hardware available (to control production line) • Changes made to library modules used • Change in relevant legislation Max 3										
1(b)(ii)	Perfective // Corrective										
1(c)	1 mark for each correct row <table border="1"> <thead> <tr> <th>Expression</th><th>Data type</th></tr> </thead> <tbody> <tr> <td>RIGHT (MachineCode, 4)</td><td>STRING</td></tr> <tr> <td>Speed * 2.5</td><td>REAL</td></tr> <tr> <td>NOT Status</td><td>BOOLEAN</td></tr> <tr> <td>IS_NUM(Check)</td><td>BOOLEAN</td></tr> </tbody> </table>	Expression	Data type	RIGHT (MachineCode, 4)	STRING	Speed * 2.5	REAL	NOT Status	BOOLEAN	IS_NUM(Check)	BOOLEAN
Expression	Data type										
RIGHT (MachineCode, 4)	STRING										
Speed * 2.5	REAL										
NOT Status	BOOLEAN										
IS_NUM(Check)	BOOLEAN										
1(d)(i)	Two / 2										

1(d)(ii)	1000
1(d)(iii)	DECLARE Product : ARRAY [0:99, 0:9] OF INTEGER 1 mark for correct upper and lower bound [0:99, 0:9] 1 mark for all other parts of declaration <u>DECLARE Product : ARRAY [0:99, 0:9] OF INTEGER</u>

1(a)(i)	MP1 When a task/module is repeated / reused / called // performed in several places MP2 A specific task can be identified /coded as a module / subroutine / procedure / function. MP3 Reduces complexity of <u>program / code</u> // Program / code is simplified MP4 Module / subroutine / procedure / function already available. MP5 <u>Testing</u> / <u>debugging</u> / <u>maintenance</u> is easier Max 2															
1(a)(ii)	MP1 Local variables are used within a module // are accessible only within the module (in which they are declared) MP2 Global variables are accessible to all parts of the program / MP3 Local variables use memory only while the module in is executing // Global variables use memory for the whole time the program is running Max 1															
1(a)(iii)	MP1 The same variable name can be reused in other parts of the program MP2 Using local variables makes modules self-contained // cannot accidentally change the same identifier outside of the module MP3 Using local variables aids modularisation. MP4 Memory allocated to local variables can be reused when module not in use Max 2															
1(b)	Mark as follows: <table><tr><th>Expression</th><th>Evaluates to</th><th>Mark</th></tr><tr><td>CHR (68)</td><td>'D'</td><td>1</td></tr><tr><td>MONTH (04/02/2025) // -2 + DAY (04/02/2025) // -2023 + YEAR (04/02/2025)</td><td>2</td><td>1</td></tr><tr><td>NOT TRUE // FALSE = TRUE</td><td>FALSE</td><td>1</td></tr><tr><td>STR_TO_NUM (MID ("Court1.4Upper", 6, 3))</td><td>1.4</td><td>2</td></tr></table> Mark Row 4 as follows: STR_TO_NUM 1 mark MID ("Court1.4Upper", 6, 3) 1 mark	Expression	Evaluates to	Mark	CHR (68)	'D'	1	MONTH (04/02/2025) // -2 + DAY (04/02/2025) // -2023 + YEAR (04/02/2025)	2	1	NOT TRUE // FALSE = TRUE	FALSE	1	STR_TO_NUM (MID ("Court1.4Upper", 6, 3))	1.4	2
Expression	Evaluates to	Mark														
CHR (68)	'D'	1														
MONTH (04/02/2025) // -2 + DAY (04/02/2025) // -2023 + YEAR (04/02/2025)	2	1														
NOT TRUE // FALSE = TRUE	FALSE	1														
STR_TO_NUM (MID ("Court1.4Upper", 6, 3))	1.4	2														

1(c)		
	Variable	Data type
	MemberCount	INTEGER
	TotalTakings	REAL
	BookingConfirmed	BOOLEAN
	MemberDOB	DATE
Mark as follows: 1 mark per row		

1(a)	One mark for each mark point MP1 Use of correct <code>Flight1</code> variable with given field names MP2 Correct assignments of four string data values MP3 Correct assignment of date data value Example answer <pre>Flight1.FlightNumber ← "SB2789" Flight1.Destination ← "Dublin" Flight1.FlightDate ← 30/07/2025 Flight1.Gate ← "N03" Flight1.Airline ← "Cambridge Airways"</pre>
1(b)(i)	One mark per mark point MP1 <code>TYPE GateID =</code> MP2 <code>(N01, N02, N03, W01, W02, W03, W04)</code> Example answer <pre>TYPE GateID = (N01, N02, N03, W01, W02, W03, W04)</pre>
1(b)(ii)	<code>DECLARE Gate : GateID</code>

1(a)	One mark per point: Error: The calculation of <code>TaxPayable</code> is incorrect Correction: <code>TaxPayable ← (ItemCost * TaxRate) / 100</code>						
1(b)(i)	Use constants (to represent the tax rate values)						
1(b)(ii)	One mark per bullet point (or equivalent to max 3): Tax rates are entered once only - Avoids / Minimise (input) error(s) / changing the Tax rates accidentally // avoids different values for tax rates at different points in the program - When required, the constant value (representing a tax rate) is changed (once) // Easier to maintain / update the program (when the tax rates change) - Makes the program / code easier to understand						
1(c)	One mark per row: <table border="1"> <thead> <tr> <th>Variable name</th><th>Data type</th></tr> </thead> <tbody> <tr> <td>HighRate</td><td>Boolean</td></tr> <tr> <td>TaxPayable</td><td>Real</td></tr> </tbody> </table>	Variable name	Data type	HighRate	Boolean	TaxPayable	Real
Variable name	Data type						
HighRate	Boolean						
TaxPayable	Real						
1(d)	OTHERWISE						
4(a)	One mark per point: Type: Run-time error Cause: A divide by zero operation is attempted // Attempt to convert a non-numeric string to a number						
4(b)	Answers include: Reason: - This part of the algorithm performs a specific task // if the check or calculation is changed it is changed only once - A (similar) subroutine is already available // Library routine is already available - The program is simplified / easier to understand / easier to design / code / test / debug / maintain Max 1 marks for 'Reason' Avoided: - Unnecessary code duplication - Errors caused by differences where several copies of the check and calculation exist Max 1 marks for 'Avoided'						

4(c)

Example solution:

```

FUNCTION Evaluate(NumStr1, NumStr2 : STRING) RETURNS Result
    DECLARE RetVal : Result
    DECLARE Num1, Num2 : REAL

    RetVal.Done ← FALSE

    IF IS_NUM(NumStr1) = TRUE AND IS_NUM(NumStr2) = TRUE THEN
        Num1 ← STR_TO_NUM(NumStr1)
        Num2 ← STR_TO_NUM(NumStr2)
        IF Num2 <> 0 THEN
            RetVal.Value ← (Num1 / Num2)
            RetVal.Done ← TRUE
        ENDIF
    ENDIF

RETURN RetVal
ENDFUNCTION

```

Note that order of parameters is not specified so divisor could be either parameter.

Mark as follows:

- 1 Function heading, parameters, ending
- 2 ... Correct return type of Result
- 3 Local declaration of type Result
- 4 Attempt at using IS_NUM() to check **both** parameters/strings
- 5 Attempt at using of STR_TO_NUM() to convert **both** parameters/strings
- 6 Check if the divisor, is non-zero ...
- 7 ... **and** if so correct calculation **and** assignment to Value field
- 8 Return variable of type Result having assigned both fields

Max 6 marks

1(a)

Pseudocode example	Selection	Iteration	Subroutine
FOR Index ← 1 TO 3 IF Safe[Index] = TRUE THEN Flap[Index] ← 0 ENDIF NEXT Index	✓	✓	
CASE OF Compound(3)	✓		✓
REPEAT UNTIL AllDone() = TRUE		✓	✓
WHILE Result[3] <> FALSE		✓	

One mark per row

1(b)

Variable	Example data value	Data type
Available	TRUE	BOOLEAN
Received	"18/04/2021"	STRING
Index	100	INTEGER

One mark per row

1(c)

Expression	Evaluates to
Available AND NOT (Index > 100)	TRUE
Index MOD 30	10
NUM_TO_STR (Index + "33")	ERROR

One mark per row

1(a)	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)	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)	The names do not reflect / indicate the purpose of the variable // the names are not meaningful	
1(c)(ii)	They make the program more difficult to understand / debug / maintain	
1(c)(iii)	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	

3(a)(i)	Pseudocode: <pre> TYPE Component DECLARE Item_Num : INTEGER DECLARE Reject : BOOLEAN DECLARE Stage : CHAR DECLARE Limit_1 : REAL DECLARE Limit_2 : REAL ENDTYPE </pre> Mark as follows: 1 One mark for TYPE and ENDTYPE statements 2 One mark for Item_Num and Reject fields 3 One mark for Stage field 4 One mark for Limit fields as REAL
---------	---

3(a)(ii)	<pre> DECLARE Item : ARRAY [1:2000] OF Component // DECLARE Item : ARRAY [2000] OF Component // DECLARE Item : ARRAY [0:1999] OF Component </pre> One mark per underlined phrase
3(b)	One mark per point: 1 Allows for iteration / can use a loop to access the records / data items 2 Use of index to directly access a record in the array // Example of simplification of code e.g. use of dot notation Item[1].Stage 3 Simplifies the code / algorithm // Reduces duplication of code // Program easier to write / understand / maintain / test / debug // Data items/record easier to search / sort / manipulate