8(a)

Example solution:

```
PROCEDURE OKToBorrow(ThisStudentID : STRING)
   DECLARE Index, Count : INTEGER

   CONSTANT Max = 5
   Count ← 0
   Index ← 1 / 0

   WHILE Index <= 5000 / Index <= 4999 AND Count < Max
      IF Loan[Index].StudentID = ThisStudentID AND __
                                 Loan[Index].OnLoan =
TRUE THEN
         Count ← Count + 1
      ENDIF
      Index ← Index + 1
   ENDWHILE

   IF Count < Max THEN
      OUTPUT "Student may borrow another book"
   ELSE
      OUTPUT "Student may not borrow another book"
   ENDIF

ENDPROCEDURE
```

Mark as follows:

1 Procedure heading, parameter and ending
2 Loop through **all elements** in `Loan` array
3 ... terminating if `Max` reached

4 Test for correct `StudentID` field **in a loop**
5 ... and that `OnLoan = TRUE` **in a loop**
... if so, then increment `Count` **in a loop**
6 Output an appropriate **message** in both cases **after the loop**

8(b)

Example solution:

```
FUNCTION ReturnBook(ThisStudentID, ThisBookID : STRING)
                                                RETURNS BOOLEAN
   DECLARE Index : INTEGER

   Index ← 1

   WHILE Index < 5001
      IF Loan[Index].StudentID = ThisStudentID  AND __
         Loan[Index].BookID = ThisBookID THEN
         IF Loan[Index].OnLoan = TRUE THEN // Not
                                                    returned
            Loan[Index].OnLoan  ← FALSE // Mark as
                                                  returned now
            RETURN TRUE  //Found and not already returned
         ELSE
            RETURN FALSE  // Found and already returned
         ENDIF
      ENDIF
      Index ← Index + 1
   ENDWHILE

   RETURN False // loan not found
ENDFUNCTION
```

Mark as follows:

1 Loop through **all elements** in `Loan` array
2 ...and book loan not found
3 Attempt to reference an individual data item **in a loop**
4 Correct test for `StudentID` **and** correct `BookID` **in a loop**
5 **…** If book not returned yet, then set `OnLoan` field to `FALSE`
6 RETURN TRUE if book loan Found and not already returned
7 RETURN FALSE if book loan Found and loan already returned
8 RETURN FALSE if book loan not found

Alterantive example solution – not immediate return

```
FUNCTION ReturnBook(ThisStudentID, ThisBookID : STRING)
                                                RETURNS BOOLEAN
   DECLARE Index : INTEGER
   DECLARE Found, OK : BOOLEAN

   Index ← 1
   Found ← FALSE
   OK ← TRUE
```

8(b)	WHILE Index < 5001 AND NOT Found
	  IF Loan[Index].StudentID = ThisStudentID AND __
	  Loan[Index].BookID = ThisBookID THEN
	  Found ← TRUE
	  IF Loan[Index].OnLoan = FALSE THEN // Already returned
	  OK ← FALSE
	  ELSE
	  Loan[Index].OnLoan ← FALSE // Mark as returned now
	  ENDIF
	  ENDIF
	  Index ← Index + 1
	ENDWHILE
	
	RETURN Found AND OK
	ENDFUNCTION

8(c)	Award marks for reference to following design features: Change to existing Record: 1 Store date due back as a new data item / in the loan record 2 Store date of loan **and** the loan length category as new data items / in the loan record **Max 1** New Record 3 **New** Records with Category (and Loan length field) Change to Program: 4 (Use the loan length category of a book to) calculate the date due date 5 Compare due date with today's date (to check if overdue / to calculate fine) 6 Map loan length category to length of loan / date due back **Max 1** **Max 2 marks**

1(a)(i)	DECLARE Car1 : Car
1(a)(ii)	**One** mark for each correct answer Example answer Car1.Colour ← "Blue" Car1.IntoStock ← 21/10/2025
1(b)(i)	**One** mark per mark point (**Max 2**) MP1 TYPE Body = MP2 (Convertible, Hatchback, Saloon, SUV) Example answer TYPE Body = (Convertible, Hatchback, Saloon, SUV)
1(b)(ii)	DECLARE BodyStyle : Body

2(a)	**Two** marks for working • correct calculation/application of exponent seen • correct method to find the final answer Working: Exponent = 8 + 2 + 1 = 11 // =0.111100101 x 2^{11} // =11110010100.0 (moving bp 11 places to right) Method to find the answer // 1024 + 512 + 256 + 128 + 16 + 4 **One** mark for correct answer Denary value: 1940 Example of solution using fractions:
2(b)	**One** mark per mark point (**Max 3**) MP1 correct **method** to find the binary number MP2 correct **use** of exponent MP3 correct answer in the space provided Working: 26.6875 converted to binary (0)11010.1011 // 16+8+2+0.5+0.125+0.0625 movement of binary point by 5 places **Mantissa** 0 1 1 0 1 0 1 0 1 1 **Exponent** 0 0 0 1 0 1

2(a)	**One** mark per mark point (**Max 3**) MP1 the precision of the number stored in the mantissa will be reduced MP2 the number of bits available for the exponent will **increase to 6** MP3 … this will increase the range of numbers that can be stored.
2(b)	**One** mark per mark point (**Max 3**) MP1 a process involving a calculation / the multiplication of two large numbers could take place MP2 the result might be outside of the range of values possible to store in the given system MP3 … leading to the most significant bits of the mantissa/exponent being lost (which is an overflow).

1(a)	**One** mark per mark point (**Max 1**) • correct answer • statement regarding number losing precision/rounding error **One** mark per mark point for working (**Max 2**) • number converted to binary 201.125 = 11001001.001 // 128 + 64 + 8 + 1 + 0.125 / $^1/_8$ seen • use of the exponent e.g. moving the binary point 8 places / $\times 2^8$. **Mantissa** 0 1 1 0 0 1 0 0 1 0 **Exponent** 0 0 1 0 0 0
1(b)	**One** mark per mark point (**Max 2**) • application of exponent to go from 1.010110011 to 101011.0011 // x 2^5 // movement of binary point 5 places seen • –32 + 8 + 2 + 1 + .125 + .0625 // –32 + 8 + 2 + 1 + $^1/_8$ + $^1/_{16}$ seen // –1 + ¼ + $^1/_{16}$ + $^1/_{32}$ + $^1/_{256}$ + $^1/_{512}$ // –1 + $^{179}/_{512}$ // $-^{333}/_{512}$ **One** mark for correct answer (**Max 1**) • –20.8125 // $-20^{13}/_{16}$