QUESTION 1.

5 A company creates two new websites, Site X and Site Y, for selling bicycles.

Various programs are to be written to process the sales data.

These programs will use data about daily sales made from Site X (using variable `SalesX`, Site Y (using variable `SalesY`).

Data for the first 28 days is shown below.

	`SalesDate`	`SalesX`	`SalesY`
1	03/06/2015	0	1
2	04/06/2015	1	2
3	05/06/2015	3	8
4	06/06/2015	0	0
5	07/06/2015	4	6
6	08/06/2015	4	4
7	09/06/2015	5	9
8	10/06/2015	11	9
9	11/06/2015	4	1
...			
28	01/07/2015	14	8

(a) Name the data structure to be used in a program for `SalesX`.

....[2]

Question 5 begins on page 12.

QUESTION 2.

5 A firm employs workers who assemble amplifiers. Each member of staff works a of hours each day.

The firm records the number of completed amplifiers made by each employee each day.

Management monitor the performance of all its workers.

Production data was collected for 3 workers over 4 days.

	Daily hours worked
Worker 1	5
Worker 2	10
Worker 3	10

Production data

	Worker 1	**Worker 2**	**Worker 3**
Day 1	10	20	9
Day 2	11	16	11
Day 3	10	24	13
Day 4	14	20	17

A program is to be written to process the production data.

(a) The production data is to be stored in a 2-dimensional array `ProductionData`, declared as follows:

```
DECLARE ProductionData ARRAY[1:4, 1:3] : INTEGER
```

(i) Describe **two** features of an array.

1 ....

....

2 ....

....[2]

(ii) Give the value of `ProductionData[3, 2]`.

....[1]

(iii) Describe the information produced by the expression:

`ProductionData[2, 1] + ProductionData[2, 2] + ProductionData[2, 3]`

....

....[2]

Write the **program code**. Do not attempt to include any validation checks.

Visual Basic and Pascal: You should include the declaration statements for variables.

Python: You should show a comment statement for each variable used with its data type.

Programming language ....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....[10]

QUESTION 3.

8 In this question you will need to use the given pseudocode built-in function:

`ONECHAR(ThisString : STRING, Position : INTEGER) RETURNS CHAR`
returns the single character at position `Position` (counting from the start of the string with v
from the string `ThisString`.
For example: `ONECHAR("Barcelona", 3)` returns 'r'.

(a) Give the value assigned to variable `y` by the following statement:

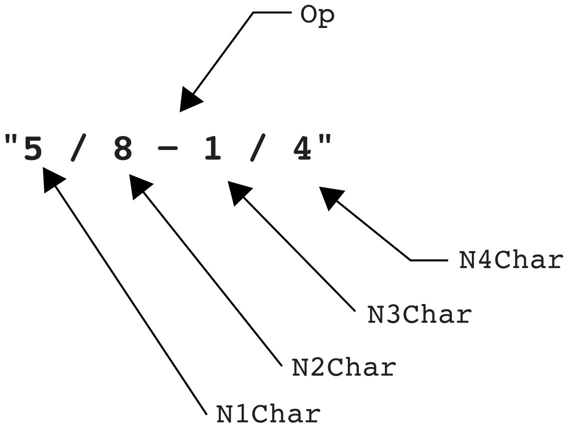
`y ← ONECHAR("San Francisco", 6)` y .... [1]

A program reads a string entered by the user. The string represents the addition or subtraction of two fractions. Each part of the fraction within the string is always a single digit only and the top digit is always less than the bottom digit.

Example strings are: `"3/8+3/5"` and `"5/8-1/4"`

The program steps are:

- the user enters the string
- the program isolates each digit and the operator
- the program computes the answer as either:
  - a fraction
  - a whole number followed by a fraction
  - a whole number
- the program displays the answer to the user



The identifier table shows the variables to be used to store the characters in the string as shown in the diagram.

Identifier	Data type	Description
FractionString	STRING	String input by user. For example: "5/8-1/4"
N1Char	CHAR	See diagram
N2Char	CHAR	See diagram
N3Char	CHAR	See diagram
N4Char	CHAR	See diagram
Op	CHAR	See diagram

(b) Study the sequence of pseudocode statements.

Show the values assigned to each variable.

```
FractionString ← "3/7+2/9"
N3Char ← ONECHAR(FractionString, 5)
Op ← ONECHAR(FractionString, 4)
```

(i) N3Char .... [1]

(ii) Op .... [1]

(iii) Complete the function call to isolate the character '9' from FractionString.

```
FractionString ← "3/7+2/9"
ONECHAR(FractionString, ...............)
```

[1]

The following additional variables are to be used by the program:

Identifier	Data type	Description
N1	INTEGER	The number value of N1Char
N2	INTEGER	The number value of N2Char
N3	INTEGER	The number value of N3Char
N4	INTEGER	The number value of N4Char
TopAnswer	INTEGER	The numerator of the fraction answer
BottomAnswer	INTEGER	The denominator of the fraction answer

(c) The following pseudocode uses these additional built-in functions:

`TONUM(ThisDigit : CHAR) RETURNS INTEGER`
returns the integer value of character `ThisDigit`
For example: `TONUM('8')` returns digit `8`.

`TOSTR(ThisNumber : INTEGER) RETURNS STRING`
returns the string value of integer `ThisNumber`
For example: `TOSTR(27)` returns `"27"`.

Study the pseudocode.

Complete the **three** dry runs for the three given values of `FractionString`.

```
OUTPUT "Enter the expression"
INPUT FractionString

// isolate each number digit and assign its number value
N1Char ← ONECHAR(FractionString, 1)
N1 ← TONUM(N1Char)
N2Char ← ONECHAR(FractionString, 3)
N2 ← TONUM(N2Char)
N3Char ← ONECHAR(FractionString, 5)
N3 ← TONUM(N3Char)
N4Char ← ONECHAR(FractionString, 7)
N4 ← TONUM(N4Char)

BottomAnswer ← N2 * N4

Op ← ONECHAR(FractionString, 4)
IF Op = '+'
   THEN
      // add fractions
      TopAnswer ← (BottomAnswer / N2) * N1 + (BottomAnswer / N4) * N3
   ELSE
      // subtract fractions
      TopAnswer ← (BottomAnswer / N2) * N1 - (BottomAnswer / N4) * N3
ENDIF

IF TopAnswer = BottomAnswer
   THEN
      OUTPUT '1'
   ELSE
      IF TopAnswer > BottomAnswer
         THEN
            TopAnswer ← TopAnswer MOD BottomAnswer
            // the & operator joins strings or character values
            OUTPUT "1 " & TOSTR(TopAnswer) & "/" & TOSTR(BottomAnswer)
         ELSE
            OUTPUT TOSTR(TopAnswer) & "/" & TOSTR(BottomAnswer)
      ENDIF
ENDIF
```

(i) `FractionString ← "2/5-3/8"`

N1	N2	N3	N4	BottomAnswer	Op	TopAnswer	OUTPUT

[2]

(ii) `FractionString ← "3/4+1/4"`

N1	N2	N3	N4	BottomAnswer	Op	TopAnswer	OUTPUT

[2]

(iii) `FractionString ← "7/9+2/3"`

N1	N2	N3	N4	BottomAnswer	Op	TopAnswer	OUTPUT

[3]

(d) The programmer writes code from the given pseudocode design. The program works, but the design is limited.

The programmer is to make amendments to the design following suggested specification changes.

(i) State the name for this type of maintenance.

....[1]

(ii) Describe **three** specification changes which will make the program more useful.

1 ....

....

2 ....

....

3 ....

....[3]

QUESTION 4.

6 Some pseudocode statements follow which use the following built-in functions:

ONECHAR(ThisString : STRING, Position : INTEGER) RETURNS CHAR
returns the single character at position Position (counting from the start of the string with v
from the string ThisString.
For example: ONECHAR("Barcelona", 3) returns 'r'.

CHARACTERCOUNT(ThisString : STRING) RETURNS INTEGER
returns the number of characters in the string ThisString.
For example: CHARACTERCOUNT("South Africa") returns 12.

(a) Study the following pseudocode statements.

Give the values assigned to variables x and y.

(i) x ← CHARACTERCOUNT("New Delhi") + 3 x .... [1]

(ii) y ← ONECHAR("Sri Lanka", 5) y .... [1]

(b) A program is to be written as follows:

- the user enters a string
- the program will form a new string with all <Space> characters removed
- the new string is output

```
NewString ← ""
INPUT InputString

j ← CHARACTERCOUNT(InputString)
FOR i ← 1 TO j
   NextChar ← ONECHAR(InputString, i)
   IF NextChar <> " "
      THEN
         // the & character joins together two strings
         NewString ← NewString & NextChar
   ENDIF
ENDFOR

OUTPUT NewString
```

(i) Complete the identifier table below.

Identifier	Data type	Description
InputString	STRING	The string value input by the user

[4]

(ii) An experienced programmer suggests this pseudocode would be bes function.

Complete the re-design of the pseudocode as follows:

The main program:

- the user enters `MyString`
- the function is called and the changed string is assigned to variable `ChangedString`

The function:

- has identifier `RemoveSpaces`
- has a single parameter
- will include the declaration for any local variables used by the function

```
// main program
INPUT MyString
ChangedString←RemoveSpaces(.........................................................................)
OUTPUT ChangedString

// function definition

FUNCTION RemoveSpaces(....................................................) RETURNS ................

    ..........................................................................................................................

    ..........................................................................................................................

    ..........................................................................................................................

    ..........................................................................................................................

    j ← CHARACTERCOUNT(InputString)
    FOR i ← 1 TO j
        NextChar ← ONECHAR(InputString, i)
        IF NextChar <> " "
            THEN
                // the & character joins together two strings
                NewString ← NewString & NextChar
        ENDIF
    ENDFOR

    ..........................................................................................................................

ENDFUNCTION
```

[7]

QUESTION 5.

6 A string-handling function has been developed. The pseudocode for this function

For the built-in functions list, refer to the **Appendix** on page 18.

```
FUNCTION SSM(String1, String2 : STRING) RETURNS INTEGER
   DECLARE n, f, x, y : INTEGER

   n ← 0
   f ← 0

   REPEAT
      n ← n + 1
      x ← n
      y ← 1
      WHILE MID(String1, x, 1) = MID(String2, y, 1)

         IF y = LENGTH(String2)
            THEN
               f ← n
            ELSE
               x ← x + 1
               y ← y + 1
         ENDIF

      ENDWHILE

   UNTIL (n = LENGTH(String1)) OR (f <> 0)

   RETURN f

ENDFUNCTION
```

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

```
SSM("RETRACE", "RAC")
```

n	**f**	**x**	**y**	**MID(String1, x, 1)**	**MID(String2, y, 1)**
0	0				

[6]

(b) **(i)** Describe the purpose of function SSM.

....

....

....

....[2]

(ii) One of the possible return values from function SSM has a special meaning.

State the value and its meaning.

Value ....

Meaning ....
[2]

(iii) There is a problem with the logic of the pseudocode. This could generate a run-time error.

Describe the problem.

....

....

....

....[2]

Appendix

Built-in functions

In each function below, if the function call is not properly formed, the function returns an error.

`MID(ThisString : STRING, x : INTEGER, y : INTEGER) RETURNS STRING`

returns the string of length `y` starting at position `x` from `ThisString`

Example: **`MID("ABCDEFGH", 2, 3)`** will return string **`"BCD"`**

`LEFT(ThisString : STRING, x : INTEGER) RETURNS STRING`

returns the leftmost `x` characters from `ThisString`

Example: **`LEFT("ABCDEFGH", 3)`** will return string **`"ABC"`**

`RIGHT(ThisString: STRING, x : INTEGER) RETURNS STRING`

returns the rightmost `x` characters from `ThisString`

Example: **`RIGHT("ABCDEFGH", 3)`** will return string **`"FGH"`**

`ASC(ThisChar : CHAR) RETURNS INTEGER`

returns the ASCII value of character `ThisChar`

Example: **`ASC('W')`** will return **`87`**

`LENGTH(ThisString : STRING) RETURNS INTEGER`

returns the integer value representing the length of string `ThisString`

Example: **`LENGTH("Happy Days")`** will return **`10`**

String operator

`&` operator

concatenates (joins) two strings

Example: **`"Summer" & " " & "Pudding"`** produces **`"Summer Pudding"`**

BLANK PAGE

QUESTION 6.

6 A string-handling function has been developed.

For the built-in functions list, refer to the **Appendix** on the last page.

The pseudocode for this function is shown below.

```
FUNCTION SF(ThisString : STRING) RETURNS STRING
   DECLARE x          : CHAR
   DECLARE NewString  : STRING
   DECLARE Flag       : BOOLEAN
   DECLARE m, n       : INTEGER

   Flag ← TRUE
   NewString ← ""
   m ← LENGTH(ThisString)

   FOR n ← 1 TO m

      IF Flag = TRUE
         THEN
            x ← UCASE(MID(ThisString, n, 1))
            Flag ← FALSE
         ELSE
            x ← LCASE(MID(ThisString, n, 1))
      ENDIF

      NewString ← NewString & x

      IF x = " "
         THEN
            Flag ← TRUE
      ENDIF

   ENDFOR

   RETURN NewString
ENDFUNCTION
```

(a) **(i)** Complete the trace table below by performing a dry run of the function when it is called as follows:

`SF("big BEN")`

n	x	Flag	m	NewString

[4]

(ii) Describe the purpose of function `SF`.

....

....

....

....[2]

(b) Test data must be designed for the function `SF`.

(i) State what happens when the function is called with an empty string.

....

....[1]

(ii) The function should be thoroughly tested.

Give **three** examples of non-empty strings that may be used.

In each case explain why the test string has been chosen.

String ....

Explanation ....

....

String ....

Explanation ....

....

String ....

Explanation ....

....

[3]

QUESTION 7.

2 A sensing device sends bit values to a computer along data channels.

- Channel 1 transmits a sequence of binary values from a sensor
- Channel 2 transmits at regular intervals to indicate whether the sensor is switched o
  - 0 indicates switched off
  - 1 indicates switched on

A program tests the bits received from the sensing device.

A program reads the signal from Channel 2 after every six values from Channel 1.

A built-in function `READ(<ChannelNumber>)` reads a value from the specified channel.

Pseudocode for the program is as follows:

```
BitCount ← 0
Status2 ← READ(2)
WHILE Status2 = 1

  FOR ReadingCount ← 1 TO 6
    ThisBit ← READ(1)
    IF ThisBit = 1
      THEN
        BitCount ← BitCount + 1
    ENDIF
    IF BitCount = 5
      THEN
        OUTPUT "Error - Investigate"
        BitCount ← 0
    ENDIF
  ENDFOR

  Status2 ← READ(2)
ENDWHILE
```

(a) Trace the execution of the program for the following sequence of bits.

Channel 1		1	0	1	1	1	0		1	1	0	0	1	1	
Channel 2	1							1							0

`Status2`	`ReadingCount`	`ThisBit`	`BitCount`	OUTPUT
			0	
1	1	1	1	
	2			

[7]

(b) Identify the following constructs in the given program, using line numbers.

For multi-line constructs give the first line number only.

Construct	**Line number**
Assignment	
Selection	
Iteration	

[3]

QUESTION 8.

5 A team keeps a record of the scores made by each of their eight players in a num

The data in the two tables below shows:

- the scores of the eight players after twenty games
- the eight player names.

	1	2	3		8
1	12	17	67		31
2	35	82	44		29
3	61	39	80		17
4	81	103	21		11
5	56	0	98		4
…					
19	45	6	81		77
20	12	11	3		6

1	Vorma
2	Ravi
3	Chada
4	Nigam
5	Bahri
6	Smith
7	Goyal
8	Lata

The team wants a computer program to input and record the player data.

(a) A programmer designs the following pseudocode for the input of a player's score from one game.

```
INPUT GameNumber
INPUT PlayerNumber
INPUT PlayerGameScore
PlayerScore[GameNumber, PlayerNumber] ← PlayerGameScore
```

Describe the data structure the programmer has used for the storage of all player scores.

.... [2]

(b) The player names are permanently stored in a text file `NAMES.TXT`, with on
line. The player names will be read by the program and stored in a 1D array.

The design given in **part (a)** will be expanded so that the user is prompted for th
name instead of the player number. Step `02` now becomes:

```
02.1    Read the player names from file NAMES.TXT into the array PlayerName
02.2    INPUT ThisPlayerName
02.3    Search the PlayerName array for ThisPlayerName to find the PlayerNumber
```

(i) State the computing term for the expansion of one or more steps in the original design.

.... [1]

(ii) Write the **program code** for step `02.1`

Visual Basic and Pascal: You should include the declaration statements for variables.
Python: You should show a comment statement for each variable used with its data type.

Programming language ....

....

....

....

....

....

....

....

....

....

....

....

.... [4]

(iii) Program code is to be designed and written for step `02.3`

The program will use these identifiers:

Identifier	Data type	Description
`PlayerName`	`ARRAY[1 : 8] OF STRING`	Stores the player names (read from the file)
`ThisPlayerName`	`STRING`	Input by the user (step `02.2`)
`Found`	`BOOLEAN`	Flags when `ThisPlayerName` is found when searching the `PlayerName` array
`i`	`INTEGER`	Array index

Write **program code** to carry out the linear search for step `02.3`

There is no requirement to declare or comment on variables used.

Programming language ....

....

....

....

....

....

....

....

....

....

....

....

.... [4]

(c) The team wants the program to produce a report, with the following specifica

The program outputs the total number of player scores that are:

- 50 and over but less than 100
- 100 or higher.

You can assume that before the section runs, the program has assigned all eight player scores to the `PlayerScore` data structure.

A first attempt at the pseudocode is shown below:

```
01 Total50 ← 0
02 Total100 ← 0
03 FOR PlayerIndex  ← 1 TO 8
04   FOR GameIndex ← 1 TO 20
05     IF PlayerScore[GameIndex, PlayerIndex] > 100
06       THEN
07         Total100 ← Total100 + 1
08       ELSE
09         IF PlayerScore[GameIndex, PlayerIndex] > 50
10           THEN
11             Total50 ← Total50 + GameIndex
12         ENDIF
13     ENDIF
14   ENDFOR
15 ENDFOR
16 OUTPUT Total50
17 OUTPUT Total100
```

(i) Describe the control structure used in lines `03` and `04` and lines `14` and `15`.

....

....

.... [2]

(ii) Consider the following two statements.

Write either TRUE **or** FALSE next to each statement.

Statement	**TRUE or FALSE**
The pseudocode considers all the scores for a player, before progressing to the next player.	
The pseudocode considers all scores in a game, before progressing to the next game.	

[1]

(iii) The programmer has made logic errors in the design.

State a line number at which an error occurs.

Explain the error or write the corrected pseudocode statement.

Line number ....

Explanation ....

.... [1]

QUESTION 1.

6 A firm employs five staff who take part in a training programme. Each memb complete a set of twelve tasks which can be taken in any order. When a me successfully completes a task, this is recorded.

A program is to be produced to record the completion of tasks for the five members of staff.

To test the code, the programmer makes the program generate test data.

The program generates pairs of random numbers:

- the first, in the range, 1 to 5 to represent the member of staff
- the second, in the range, 1 to 12 to represent the task

Each pair of numbers simulates the completion of one task by one member of staff.

(a) Explain why the generation of 60 (5 staff x 12 tasks) pairs of random numbers will not simulate all tasks completed by all staff.

....

....

....[2]

(b) Data is currently recorded manually as shown.

Staff number	**Task number**											
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
1												
2												
3				✓								
4												
5								✓				

The table shows that two members of staff have each successfully completed one task.

The program must use a suitable data structure to store, for all staff:

- tasks successfully completed
- tasks not yet successfully completed

The program will output the staff number and task number in the order in which tasks are completed.

The program design in pseudocode is produced as follows:

```
DECLARE StaffNum          : INTEGER
DECLARE TaskNum           : INTEGER
DECLARE ..............................................................................................
DECLARE NewStaffTask      : BOOLEAN

CALL InitialiseTaskGrid
Completed ← 0
WHILE Completed <> 60
   NewStaffTask ← FALSE
   WHILE NewStaffTask = FALSE
      StaffNum ← RANDOM(1,5)        //generates a random number
      TaskNum ← RANDOM(1,12)        //in the given range
      IF TaskGrid[StaffNum, TaskNum] = FALSE
         THEN
            TaskGrid[StaffNum, TaskNum] ← TRUE
            NewStaffTask ← TRUE
            OUTPUT StaffNum, TaskNum
      ENDIF
   ENDWHILE
   Completed ← Completed + 1
ENDWHILE
OUTPUT "Staff Task Count", Completed

// end of main program

PROCEDURE InitialiseTaskGrid()
   DECLARE i : INTEGER
   DECLARE j : INTEGER
   FOR i ← 1 TO 5
      FOR j ← 1 TO 12
         TaskGrid[i, j] ← FALSE
      ENDFOR
   ENDFOR
ENDPROCEDURE
```

Study the pseudocode and answer the questions below.

Give the line number for:

(i) The declaration of a `BOOLEAN` global variable. ....

(ii) The declaration of a local variable. .... [1

(iii) The incrementing of a variable used as a counter, but not to control a 'count controlled' loop.

.... [1]

(iv) A statement which uses a built-in function of the programming language.

.... [1]

(c) (i) State the number of parameters of the `InitialiseTaskGrid` procedure.

.... [1]

(ii) Copy the condition which is used to control a 'pre-condition' loop.

....[1]

(iii) Explain the purpose of lines `13` – `18`.

....

....

....

....

....

....[3]

(iv) Give the global variable that needs to be declared at line `03`.

....[2]

(d) Line 17 in the pseudocode outputs the staff number and the task number.

A new requirement is to display the name of the member of staff given in the table.

Write a CASE structure using variable `StaffNum`.

Assign to a new variable `StaffName` the appropriate staff name.

Staff number	St
1	Sadiq
2	Smith
3	Ho
4	Azmah
5	Papadopoulos

[4]

Question 7 begins on page 14.

QUESTION 2.

4 The standard pack of playing cards has four suits – called Clubs, Diamonds, Hea
Each card has a value shown by its number or a name: 1 (Ace), 2, 3, … 10, 11 (Jack,
13 (King). The pack of cards has one combination for each suit and value.

A program is to be written which simulates a magician dealing all 52 cards from the card pac

The program generates pairs of random numbers:

- the first, in the range 1 to 4, to represent the suit
- the second, in the range 1 to 13, to represent the card value

(a) Explain why the generation of 52 (4 suits x 13 card values) pairs of random numbers will not simulate the dealing of the complete pack.

....

....

....[2]

(b) A representation of dealing out the cards is shown below:

Suit number	**Card value**												
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
1 (Clubs)	F	F	F	F	F	F	F	F	F	F	T	F	F
2 (Diamonds)	F	F	F	F	F	F	F	F	F	F	F	F	F
3 (Hearts)	F	F	T	F	F	F	F	F	F	F	F	F	F
4 (Spades)	F	F	F	F	F	F	F	F	F	F	F	F	F

The table shows two cards have been dealt so far; the 3 of Hearts and the Jack of Clubs.

When each card is dealt, the appropriate cell changes from F to T.

The program will output the suit and the card value in the order in which the cards are dealt.

Question 4(b) continues on page 8.

The program design in pseudocode is produced as follows:

```
DECLARE SuitNum      : INTEGER
DECLARE CardValue    : INTEGER
DECLARE DealCount    : INTEGER
DECLARE NewCard      : BOOLEAN
DECLARE CardPack ..................................................

CALL InitialiseCardPack
DealCount ← 0
WHILE DealCount <> 52
   NewCard ← FALSE
   WHILE NewCard = FALSE
      SuitNum ← RANDOM(1,4)     // generates a random number
      CardValue ← RANDOM(1,13) // in the range given
      IF CardPack[SuitNum, CardValue] = FALSE
         THEN
            CardPack[SuitNum, CardValue] ← TRUE
            NewCard ← TRUE
            OUTPUT SuitNum, CardValue
      ENDIF
   ENDWHILE
   DealCount ← DealCount + 1
ENDWHILE

// end of main program

PROCEDURE InitialiseCardPack
   DECLARE i : INTEGER
   DECLARE j : INTEGER
   FOR i ← 1 TO 4
      FOR j ← 1 TO 13
         CardPack[i, j] ← FALSE
      ENDFOR
   ENDFOR
ENDPROCEDURE
```

Study the pseudocode and answer the questions below:

Give the line number for:

(i) A statement which marks the end of a count controlled loop.

....[1]

(ii) The declaration of a local variable.

....[1]

(iii) The initialisation of a variable used as a counter, but not to control a 'count controlled' loop.

....[1]

(iv) A statement which uses a built-in function of the programming language.

....[1]

(c) Give the number of procedures used by the pseudocode.

....[1]

(d) Copy the condition which is used to control a 'pre-condition' loop.

....[1]

(e) Explain the purpose of lines `14` – `19` in the design.

....

....

....

....

....

....[2]

(f) Complete the declaration of the global variable at line `05`.

`05 DECLARE CardPack` ....[1]

(g) Line `18` in the design shows which new card is dealt each time.

When an Ace, Jack, Queen or King is dealt, the output displays the number for th
the name of the card.

Card value	**Card name**
1	Ace
11	Jack
12	Queen
13	King

A new requirement is to display the name of the card, where appropriate.

Write a CASE structure using variable `CardValue`.

Assign to a new variable `CardName` either:

- the card value (2, 3, 4, 5, 6, 7, 8, 9 or 10)
- or where appropriate, the card name Ace, Jack, Queen or King

....

....

....

....

....

....

....

....[4]

QUESTION 3.

6 A multi-user computer system makes use of passwords.

To be valid, a password must comply with the following rules:

- at least two lower-case alphabetic characters
- at least two upper-case alphabetic characters
- at least three numeric characters
- alpha-numeric characters only

A function, `ValidatePassword`, is needed to check that a given password follows these rules. This function takes a string, `Pass`, as a parameter and returns a Boolean value:

- `TRUE` if `Pass` contains a valid password
- `FALSE` otherwise.

(a) Write **program code** to implement the new function `ValidatePassword`.

Visual Basic and Pascal: You should include the declaration statements for variables.
Python: You should show a comment statement for each variable used with its data type.

Programming language ....

Program code

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....[10]

(b) **(i)** The function will be tested.

Give a valid string to check that the function returns `TRUE` under the correct c

String1: ....

Modify the valid string given for String1 to test each rule separately.

Explain your choice in each case.

String2: ....

Explanation: ....

....

....

String3: ....

Explanation: ....

....

....

String4: ....

Explanation: ....

....

....

String5: ....

Explanation: ....

....

....

[5]

(ii) When testing a module, it is necessary to test all possible paths through the code.

State the name given to this type of testing.

....[1]

(iii) A program consisting of several modules may be tested using a pro
stub testing.

Explain this process.

....

....

....

....[2]

Appendix

Built-in functions (pseudocode)

In each function, if the function call is not properly formed, the function returns an error.

```
MID(ThisString : STRING, x : INTEGER, y : INTEGER) RETURNS STRING
```

returns string of length `y` starting at position `x` from `ThisString`.
Example: **`MID("ABCDEFGH", 2, 3)`** returns string **`"BCD"`**

```
LENGTH(ThisString : STRING) RETURNS INTEGER
```

returns the integer value representing the length of string `ThisString`.
Example: **`LENGTH("Happy Days")`** returns **`10`**

```
LEFT(ThisString : STRING, x : INTEGER) RETURNS STRING
```

returns leftmost `x` characters from `ThisString`.
Example: **`LEFT("ABCDEFGH", 3)`** returns string **`"ABC"`**

```
RIGHT(ThisString: STRING, x : INTEGER) RETURNS STRING
```

returns rightmost `x` characters from `ThisString`.
Example: **`RIGHT("ABCDEFGH", 3)`** returns string **`"FGH"`**

```
LCASE(ThisChar : CHAR) RETURNS CHAR
```

returns the character value representing the lower case equivalent of `ThisChar`.
If `ThisChar` is not an upper-case alphabetic character then it is returned unchanged.
Example: **`LCASE('W')`** returns **`'w'`**

```
MOD(ThisNum : INTEGER, ThisDiv : INTEGER) RETURNS INTEGER
```

returns the integer value representing the remainder when `ThisNum` is divided by `ThisDiv`.
Example: **`MOD(10,3)`** returns **`1`**

```
DIV(ThisNum : INTEGER, ThisDiv : INTEGER) RETURNS INTEGER
```

returns the integer value representing the whole number part of the result when `ThisNum` is divided by `ThisDiv`.
Example: **`DIV(10,3)`** returns **`3`**

Operators (pseudocode)

Operator	Description
`&`	Concatenates (joins) two strings. Example: **`"Summer" & " " & "Pudding"`** produces **`"Summer Pudding"`**
`AND`	Performs a logical **`AND`** of two Boolean values. Example: **`TRUE AND FALSE`** produces **`FALSE`**
`OR`	Performs a logical **`OR`** of two Boolean values. Example: **`TRUE OR FALSE`** produces **`TRUE`**

BLANK PAGE

QUESTION 4.

1 (a) (i) Procedural high-level languages usually support different data types.

Give an appropriate data type for each data value in the following table.

Data value	Data type
`27`	
`"27"`	
`"27.3"`	
`TRUE`	
`27/3/2015`	
`27.3`	

[6]

(ii) State an appropriate data structure to store the individual test scores for a class of students.

....[1]

(iii) Describe how characters are represented using the ASCII character set.

....

....

....

....[2]

(b) Functions and procedures are subroutines.

Explain why you should use subroutines when designing a program solution.

....

....

....

....[2]

(c) The following pseudocode is an example of nested `IF` statements.

```
IF MyVar = 1
   THEN
      CALL Proc1()
   ELSE
      IF MyVar = 2
         THEN
            CALL Proc2()
         ELSE
            IF MyVar = 3
               THEN
                  CALL Proc3()
               ELSE
                  OUTPUT "Error"
            ENDIF
      ENDIF
ENDIF
```

Use **pseudocode** to write a `CASE` statement with the same functionality.

....

....

....

....

....

....

....

....[4]

(d) Program coding is a transferable skill.

You are given program code written in a high-level language that you have not studied.

State **two** different features of the code that you should be able to recognise.

1 ....

....

2 ....

....

[2]

QUESTION 5.

1 (a) (i) Procedural high-level languages usually support different data types.

Give an appropriate data type for each data value in the following table:

Data value	Data type
FALSE	
03/03/2013	
35	
"INTEGER"	
3.5	
"35"	

[6]

(ii) The following is a declaration in a high-level language:

```
DEFINE MyGrade[1 to 100]
```

State the data structure of variable MyGrade.

.... [1]

(iii) An experienced programmer is presented with program code in an unfamiliar high-level language.

State **two** features of the code that the programmer should be able to recognise.

1 ....

....

2 ....

....

[2]

(b) **(i)** In the ASCII character set 'A' is represented by the value 65. The values other characters of the alphabet follow in sequence, so 'B' is represented by and so on.

The following table represents consecutive memory locations. Each memory lo stores one byte.

Complete the table to show how the string "CAGE" may be stored in memory using the ASCII set.

Address	Data
`100`	
`101`	
`102`	
`103`	
`104`	
`105`	

[2]

(ii) In a high-level language, a `LENGTH` function is used to return the number of characters in a string.

Explain what is stored in addition to the string characters to allow this function to determine this number.

....

....

....

.... [2]

(c) Functions and procedures are subroutines.

Explain why parameters are used with subroutines.

....

....

....

....

....

.... [3]

(d) The following pseudocode is an example of a `CASE` structure.

```
CASE OF MyMark
   75 to 100: MyGrade ← "Distinction"
   35 to 74: MyGrade ← "Pass"
   0 to 34: MyGrade ← "Fail"
   OTHERWISE: OUTPUT "Invalid value entered"
ENDCASE
```

(i) Describe what will happen if the pseudocode is tested when `MyMark` has the following values:

27 ....

....

101 ....

....
[2]

(ii) Use **pseudocode** to write an `IF` statement with the same functionality.

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

.... [5]

Question 2 begins on the next page.

QUESTION 6.

1 **(a)** The following table contains statements written in pseudocode.

Show the type of programming construct each statement represents.

Put a tick (✓) in the appropriate column for each statement.

Statement	Assignment	Selection	Repetition (Iteration)
`CASE OF TempSensor1`			
`ELSE`			
`REPEAT`			
`ENDFOR`			
`DayNumber ← DayNumber + 1`			
`Error ← TRUE`			

[6]

(b) **(i)** The following table contains statements written in pseudocode.

Give the most appropriate data type for the variable used in each statement.

Statement	Data type
`Revision ← 500`	
`FuelType ← 'P'`	
`MinValue ← -6.3`	
`ServiceDue ← FALSE`	
`ModelRef ← "W212DEC15"`	

[5]

(ii) The following table contains statements written in pseudocode.

Complete the table by evaluating each expression using the values from part **(b)(i)**.

If any expression is invalid, write "ERROR" in the **Evaluates to** column.

For the built-in functions list, refer to the **Appendix** on page 16.

Expression	Evaluates to
`"Month: " & MID(ModelRef, 5, 3)`	
`INT(MinValue * 2)`	
`ASC(Revision)`	
`Revision > 500`	
`ServiceDue = TRUE OR FuelType = 'P'`	

[5]

QUESTION 7.

6 Account information for users of a library is held in one of two text files; `UserLis` `UserListNtoZ.txt`

The format of the data held in the two files is identical. Each line of the file is stored as a s contains an account number, name and telephone number separated by the asterisk cha ('*') as follows:

```
<Account Number>'*'<Name>'*'<Telephone Number>
```

An example of one line from the file is:

```
    "GB1234*Kevin Mapunga*07789123456"
```

The account number string may be **six** or **nine** characters in length and is **unique for each person**. It is made up of alphabetic and numeric characters only.

An error has occurred and the same account number has been given to different users in the two files. There is **no** duplication of account numbers **within each individual file**.

A program is to be written to search the two files and to identify duplicate entries. The account number of any duplicate found is to be written to an array, `Duplicates`, which is a 1D array of 100 elements of data type `STRING`.

The program is to be implemented as several modules. The outline description of three of these is as follows:

Module	Outline description
`ClearArray()`	• Initialise the global array `Duplicates`. Set all elements to the empty string.
`FindDuplicates()`	• Read each line from the file `UserListAtoM.txt` ○ Check whether the account number appears in file `UserListNtoZ.txt` using `SearchFileNtoZ()` ○ If the account number does appear then add the account number to the array. • Output an error message and exit the module if there are more duplicates than can be written to the array.
`SearchFileNtoZ()`	• Search for a given account number in file `UserListNtoZ.txt` ○ If found, return `TRUE`, otherwise return `FALSE`

(a) State **one** reason for storing data in a file rather than in an array.

....

.... [1]

(b) Write **program code** for the module `SearchFileNtoZ()`.

Visual Basic and Pascal: You should include the declaration statements for variab
Python: You should show a comment statement for each variable used with its data t

Programming language ....

Program code

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

.... [7]

(c) Write **pseudocode** for the module `FindDuplicates()`.

The module description is given in the table on page 12.

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

.... [8]

(d) `ClearArray()` is to be modified to make it general purpose. It will be used to initialise any 1D array of data type `STRING` to any value.

It will now be called with three parameters as follows:

1. The array
2. The number of elements
3. The initialisation string

You should assume that the lower bound is 1.

(i) Write **pseudocode** for the modified `ClearArray()` procedure.

....

....

....

....

....

....

....

.... [3]

(ii) Write **program code** for a statement that calls the modified `ClearArray()` procedure to clear the array `Duplicates` to `"Empty"`.

Programming language ....

Program code

....

....

[2]

Appendix

Built-in functions (pseudocode)

Each function returns an error if the function call is not properly formed.

`MID(ThisString : STRING, x : INTEGER, y : INTEGER) RETURNS STRING`
returns a string of length `y` starting at position `x` from `ThisString`

Example: `MID("ABCDEFGH", 2, 3)` returns `"BCD"`

`LENGTH(ThisString : STRING) RETURNS INTEGER`
returns the integer value representing the length of `ThisString`

Example: `LENGTH("Happy Days")` returns `10`

`LEFT(ThisString : STRING, x : INTEGER) RETURNS STRING`
returns leftmost `x` characters from `ThisString`

Example: `LEFT("ABCDEFGH", 3)` returns `"ABC"`

`RIGHT(ThisString: STRING, x : INTEGER) RETURNS STRING`
returns rightmost `x` characters from `ThisString`

Example: `RIGHT("ABCDEFGH", 3)` returns `"FGH"`

`INT(x : REAL) RETURNS INTEGER`
returns the integer part of `x`

Example: `INT(27.5415)` returns `27`

`NUM_TO_STRING(x : REAL) RETURNS STRING`
returns a string representation of a numeric value.
Note: This function will also work if `x` is of type `INTEGER`

Example: `NUM_TO_STRING(87.5)` returns `"87.5"`

`STRING_TO_NUM(x : STRING) RETURNS REAL`
returns a numeric representation of a string.
Note: This function will also work if `x` is of type `CHAR`

Example: `STRING_TO_NUM("23.45")` returns `23.45`

`ASC(ThisChar : CHAR) RETURNS INTEGER`
returns the ASCII value of `ThisChar`

Example: `ASC('A')` returns `65`

`CHR(x : INTEGER) RETURNS CHAR`
returns the character whose ASCII value is `x`

Example: `CHR(87)` returns `'W'`

`UCASE(ThisChar : CHAR) RETURNS CHAR`
returns the character value representing the upper case equivalent of `ThisChar`
If `ThisChar` is not a lower case alphabetic character, it is returned unchanged.

Example: `UCASE('a')` returns `'A'`

Operators (pseudocode)

Operator	Description
&	Concatenates (joins) two strings Example: `"Summer" & " " & "Pudding"` produces `"Summer Pudding"`
AND	Performs a logical `AND` on two Boolean values Example: `TRUE AND FALSE` produces `FALSE`
OR	Performs a logical `OR` on two Boolean values Example: `TRUE OR FALSE` produces `TRUE`

BLANK PAGE

BLANK PAGE

QUESTION 8.

4 The following pseudocode algorithm checks whether a string is a valid email addr

```
FUNCTION Check(InString : STRING) RETURNS BOOLEAN

   DECLARE Index : INTEGER
   DECLARE NumDots : INTEGER
   DECLARE NumAts : INTEGER
   DECLARE NextChar : CHAR
   DECLARE NumOthers : INTEGER

   NumDots ← 0
   NumAts ← 0
   NumOthers ← 0

   FOR Index ← 1 TO LENGTH(InString)

      NextChar ← MID(InString, Index, 1)
      CASE OF NextChar
         '.': NumDots ← NumDots + 1
         '@': NumAts ← NumAts + 1
         OTHERWISE NumOthers ← NumOthers + 1
      ENDCASE

   ENDFOR

   IF (NumDots >= 1 AND NumAts = 1 AND NumOthers > 5)
      THEN
         RETURN TRUE
      ELSE
         RETURN FALSE
   ENDIF

ENDFUNCTION
```

(a) Describe the validation rules that are implemented by this pseudocode. Refer **only** to the contents of the string and **not** to features of the pseudocode.

....

....

....

.... [3]

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

```
Result ← Check("Jim.99@skail.com")
```

Index	**NextChar**	**NumDots**	**NumAts**	**NumOthers**

[5]

(ii) State the value returned when function `Check` is called as shown in **part (b)(i)**.

.... [1]

(c) The function `Check()` is to be tested.

State **two** different invalid string values that could be used to test the algorithm.
should test a different rule.

Justify your choices.

Value ....

Justification ....

....

....

Value ....

Justification ....

....